

初心者からプロまで必携

パソコンPCシリーズ

8001 6001

ハンドブック

●朝日新聞電子計算室編●

朝日新聞社

800円





は じ め に

世はあげてパソコン時代である。しかし、あまりの急成長のためか、パソコンのベストセラーであるNECのPC-8001ですら、まとまった機能解説のハンドブックがなかった。朝日新聞社のマイコン研修会でも、教育機器の観点から使用機種にPC-8001を選んだが、発足後半年ほどたったころ、会員の間から自然発生的に「使いやすく、適切なハンドブックが欲しい」という声が出てきた。たしかに、機能・操作の各項目について断片的な資料は出ているけれども、まとまったものはない。そこで、会員のためにハンドブックをまとめよう、ということになった。

約一年間かけて編集が終わりかけたころ、出版局プロジェクト室から、せっかくまとめたのだから単に社内用の資料に終わらせず、一般に公開して、広く社会に役立ててはどうかとのすすめがあった。「それでは初心者からプロまで、だれにでも役立つようなハンドブックを作ろう」と意気込み、新しく発売されたPCシリーズの普及版、PC-6001の機能解説も加えて、再編集したのがこのハンドブックである。初心者がどこで行き詰まるかも、マイコン研修会の研修記録から洗い出して内容に反映させた。初心者研修用のテキスト例は、自習用としても役立つはずである。PC-6001については巻末にまとめておいた。

責任上、本文・表を通じて島戸が執筆した。さらに江田、西村、杉山を加え、徹底的な討議を繰り返して改稿した。NEC技術陣からも、多数の資料と有益な意見をいただいた。

このハンドブックが、パソコンを使いこなし、機能を最大限に引き出す上でお役に立てば幸いである。

朝日新聞社電子計算室長

島戸 一臣

本書の特長・使用法

(1) コマンドおよびステートメントの説明は次の原則にしたがう。

< >で囲まれたものは使用者が指定する。

[]で囲まれたものは必要なら指定する。

(ディスク) とあるのは、ディスクが接続されているときに使える。

(2) 16進表示は次の原則にしたがう。

BASICの場合は数字の頭部に&Hをつける。

機械語やニーモニックの場合は数字の末尾にHをつける。

(3) 索引はコマンドやステートメントだけではなく、機能でも引けるようにした。

(4) 15回分のテキストは、社内研修で約400人が使ったものに、加筆訂正を加えて作った。あらかじめキーボードに慣れさせておかないと、受講者のレベルにもよるが1回2時間では苦しいかもしれない。

◇ この本は執筆者4人がワードプロセッサを使い、直接、版下を作りました。

◇ 恐れいますが、この本の内容に関するお問い合わせは、ハガキか封書でお願いいたします。ご返事は電話でいたしますので、電話番号を忘れずにお書き添えください。

あて先 ㊤104 東京都中央区築地5-3-2

朝日新聞社 出版局プロジェクト室

執筆者

島戸 一臣

1936年生まれ

朝日新聞社電子計算室長兼技術本部システム管理室長

朝日新聞社マイクロコンピューター研修会事務局長

江田 総司

1953年生まれ

朝日新聞社電子計算室員，システムリーダー

第一種情報処理技術者

西村 哲

1957年生まれ

朝日新聞社電子計算室員，第一種情報処理技術者

杉山 裕一

1958年生まれ

朝日新聞社電子計算室員，第一種情報処理技術者

表紙デザイン

東 盛太郎

目	次
---	---

1. PC-8001	6	1 1. 浮動小数点形式	4 4
キー配置図	6	内部表現を画面表示する	4 6
キーの説明	6, 7		
2. BASICの基本文法	8	1 2. USR関数	4 7
演算の優先順位	9	浮動小数点アキュムレーター	4 7
3. 操作上の注意点	9	ストリングディスクリプター	4 8
4. 文字・演算子一覧表	1 0	USR関数を使った機械語例	4 8
機能・型宣言・算術演算	1 0	画面をプリントする	4 9
論理演算・関係演算	1 1	ユーザールーチン	
5. コマンドとステートメント一覧	1 2	使用上の注意点	5 0
(1) コマンド	1 4	1 3. プリンターの使い方	5 2
(2) ステートメント	1 8	文字制御	5 3
(3) 入出力ステートメント	2 3	幅・行・位置の制御	5 4
(4) スクリーン		1 4. ミニフロッピーディスク	5 6
ステートメント	2 6	ディスクファイルの管理	5 7
図形移動	2 9	バックアップのとりかた	5 9
(5) 特殊ステートメント	3 1	フォーマットのしかた	6 0
WAITとINPの例	3 2	データファイル	6 1
6. 数値関数	3 4	シーケンシャルファイル	6 2
7. ストリング関数	3 6	ランダムファイル	6 5
8. ディスク関数	3 8	特殊な I/O	6 7
9. その他の関数	4 0	1 5. エラーメッセージ	6 9
1 0. PRINTUSING一覧表	4 2	1 6. PC-8000ラインアップ	7 3
		RAMの増設	7 6
		BANKの切りかえ	7 7
		1 7. キャラクターコード表	7 8

目次

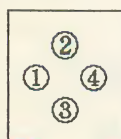
18. PRINTCHR\$ INKEY\$	80	30. 機械語ニーモニック変換表	115
19. CPUブロック図	82	31. ニーモニック機械語変換表	131
20. CPUレジスター	83	32. カリキュラム例	137
メモリーの種類	84	講習の注意点	138
21. メモリーアドレスマップ	85	33. テキスト15回分	139
22. BASICの内部型	86	34. ディスクを使った プログラムの例	155
23. ビデオRAM	88	35. PC-6001	163
アトリビュートコードの値	89	キー配置図	164
24. ターミナルモード	90	文字・演算子	166
モデム	91	コマンド・ステートメント 一覧表	166
RS-232C	91	(1) コマンド	167
IEEE-488	91	(2) ステートメント	167
USART	91	(3) 入出力ステートメント	169
25. I/Oマップ	92	(4) 一般関数	170
26. キーボードマップ	94	(5) 入出力関数	171
27. 予約語マップ	96	(6) スtring関数	171
28. 機械語プログラムの作り方	98	各機能の詳細	172
機械語プログラムの例	100	キャラクターコード表	178
29. Z80相当機械語説明表	104	メモリーアドレスマップ	180
		エラーメッセージ	181
		PC-6000ラインアップ	183
		36. 索引	184

PC-8001 キー配置図

ファンクションキーの定義
 SHIFT+f1→TIME \$ f1のみ→HT
 SHIFT+f2→KEY f2のみ→auto
 SHIFT+f3→PRINT f3のみ→goto

ファンクションキー→		f・1		f・2		f・3		f・4	
1 !	2 "	3 # ア ア	4 \$ ウ ウ	5 % エ エ	6 & オ オ	7 ' ヤ ヤ	8 (ユ ユ	9) ヨ ヨ	
ESC	Q タ	W テ	E イ	R ス	T カ	Y ン	U ナ	I ニ	
CTRL	A チ	S ト	D シ	F ハ	G キ	H ク	J マ	K ノ	
SHIFT	Z ツ	X サ	C ソ	V ヒ	B コ	N ミ	M モ		
		GRAPH	スペースキー						

(1) キー



- ① キーを押すと、①の位置の文字が出る。英字の場合は小文字。
- ② SHIFT を押してキーを押すと②の位置の文字が出る。英字の場合は大文字が出る。
- ③ カナキーを押してキーを押すと③の位置の文字が出る。
- ④ カナキーとSHIFT を押してキーを押すと④の位置の文字が出る。

(2) ESCキー

プログラムの一時停止。再開はキーボード上のどのキーを押してもよい。

(3) CTRLキー

スクリーンエディットのときに使う。このキーを押しながら次のキーを押す。

R→INSキーと同じ。

H→DELキーと同じ。

N→次の単語にカーソルを移す。

B→前の単語にカーソルを移す。

J→カーソル以後を次の行に分割する。

E→カーソルから行の終わりまでを消す。

G→BEEP

C→入力待ちのときSTOPと同じ。

I→水平タブ。

K→HOMEキーと同じ。

L→CLRキーと同じ。

M→RETURNキーと同じ。

(4) STOP 卡一

プログラムの強制停止。再開はCONTコマンド (CONT参照)

SHIFT+f4→list. SHIFT+f5→cont+RET
f4のみ→list f5のみ→run+RET

(5) CLR →押すと画面クリアー。

HOME→SHIFT+CLR カーソルが左上へ。

DEL →カーソルの左1文字を消して
ひとつ詰める。

INS →カーソルの位置に空白を挿入。

矢印→カーソルを各方向に移動。

f・4		f・5		DEL	
0 ヲ ワ		= ホ		^ ヘ	
≠		STOP		INS - 矢印	
O ラ		P セ		@ "	
〔		RETURN		〕	
L リ		+ ; レ		* : ケ	
カ		ナ		ム	
<		>		?	
・		°		/	
ネ		ル		メ	
—		ロ		SHIFT	

HOME	↓	←	INS
CLR	↑	→	DEL
7	8	9	*
4	5	6	+
1	2	3	=
0	,	.	RET

(6) RETURN ㊦—

入力の終わりに必ず押すこと。

(7) スペースキー

このキーを押すと空白がひとつとられる。

(8) カナキ一

押すとロックされてカナモードになる。解除はもう一度押す。

[illegible]

┌	┐	└	┘
≡	≠	≡	≡
└	┐	└	┘
┌	┐	└	RET

GRAPH \rightarrow
 $\frac{1}{2} -$ \perp

上の文字および図形を出すには、GRAPH キーをおしながら各キーを押す。

BASICの基本文法

- (1) BASIC言語で書かれた処理手順の命令をステートメントという。ステートメントの組み合わせでプログラムを作る。プログラムへの指令をコマンドという。ステートメントのなかにはコマンドとして働くものもある。
- (2) プログラムは小文字で入力しても大文字で入力してもかまわない。
- (3) ダイレクトモードでは電卓のように使える。この場合、行番号をつけない。

```
PRINT 3.14159*45
```

```
? SQR (3)
```

```
FOR J=1 TO 10:PRINT J*2;:NEXT J
```

プログラムモードでは、まずプログラムとして記憶する。それから実行する。
したがって、なん回でも実行できる。行の始めに必ず行番号がくる。

```
10 INPUT R
```

```
20 A=3.14159*R
```

```
30 PRINT A
```

- (4) 行番号は0から65529までの正整数でつける。1行の長さは255文字まで。行番号や空白も1字に数えられる。コロン(:)で区切ると1行に複数のステートメントが書ける。(これをマルチステートメントという)
- (5) 定数とはBASICの実行時、そのまま使える値をいう。数値定数と文字定数とがある。文字定数は引用符(" ")で囲む。

- (6) 変数とはBASICプログラムのなかで使われる値を表す名前をいう。数値変数と文字変数とがある。変数のアタマ1文字だけは英字とする。変数のなかに予約語(PRINT, IF, ONなど)や特殊文字("や:など)を含めてはいけない。変数の長さは2文字以内。2文字以上つけるとアタマ2文字で判定される。文字変数は、語尾に\$(ドルマーク)をつける。
- (7) 式とは、定数、変数、関数、文字、演算子の組み合わせで、結果が数値か文字列になるものをいう。
- (8) 数値と文字列は別々に扱う。文字列は" "で囲うこと。
- (9) 演算の優先順位は
カッコで囲まれた式→べき乗→負数→*, /→¥→MOD→+, -→条件判定→
NOT→AND→OR→XOR→IMP→EQV

操作上の注意点

- (1) スイッチを切って再度入れるときには、5秒以上待つこと。
- (2) リセットボタンを押すと初期状態にもどる。プログラムも消える。
STOPボタンを押しながらリセットボタンを押すとプログラムは消えずに、画面がクリアーされてOKと出る。
- (3) 初心者はRETURNキーを押し忘れる。前のプログラムをNEWで消さないで次のプログラムを入力することがある。

文字・演算子

	文 字	意 味
機 能	&Oまたは&	8進定数 &はアンパーサントと読む
	&H	16進定数 &H51など
	疑問符 (?)	PRINTと同じ L?とは使えない
	アポストロフィ (')	REMと同じ 1行の途中からでも使える
	ピリオド (.)	現在の行番号 たとえばIllegal Function Call in 30と出た場合にはLIST. で30行が出てくる
	コロソ (:)	1行に複数の文を入れるときの区切り記号
	セミコロソ (;)	PRINT文での区切り記号
	コンマ (,)	PRINT文を14字ずつ区切る
	引用符 (")	引用符で囲むと文字列になる "ABC"や"6"
型 宣 言	\$	ストリング (文字列) を宣言する A\$やB\$など
	%	-32768から32767の整数宣言 A%など
	!	7. 1桁の浮動小数の単精度実数宣言 A!など
	#	16. 8桁の浮動小数の倍精度実数宣言 B#など
算 術 演 算	=	代入または等価判定 C=A+Bなど
	-	負数表示または減算 -15やA-Bなど
	+	加算や文字列の結合。A+Bや"AB"+"CD"
	*	乗算 A*Bや3*2, あるいはC*4など
	/	結果が浮動小数点である除算 A/BやC/2
	^	べき乗 A^2 これはA*Aと同じ
	÷	結果が整数である除算 11÷3は3
	MOD	結果が整数である剰余 11MOD3は2

文字・演算子

	文 字	意 味
論 理 演 算	NOT	否定または1の補数 結果は整数
	AND	論理積 結果は整数
	OR	論理和 結果は整数
	XOR	排他的論理和 どちらか一方にビットが立っていれば1, 両方立っていれば0 両方とも0は0
	EQV	等価 XORの否定になる 整数
	IMP	インプリケーション Xの否定またはY X IMP YはYが1なら1 あとはXの否定 整数
関 係 演 算	=, >, < <>, >< =<, >= =>, <= など	条件判定 整数, 真なら-1, 偽なら0になる

/11
010
010

コマンドとステートメント

n, X, Y, Z は数値
S は文字列とする

一般	コマンド	AUTO CONT DELETE KEYLIST LIST LLIST MON NEW RENUM RUN TERM
	ステートメント	CLEAR DATA DEFFN DEFDBL DEFINT DEFSNG DEFSTR DIM END ERASE FOR GOSUB GOTO IF~THEN LET NEXT ON~GOSUB ON~GOTO READ REM RESTORE RETURN STOP SWAP DEFUSR
	入出力ステートメント	INPUT LINEINPUT LPRINT PRINTUSING LPRINTUSING OUT POKE PRINT WAIT
	画面ステートメント	COLOR CONSOLE GET@ LINE LOCATE PRESET PSET PUT@ WIDTH
	特殊ステートメント	BEEP ERROR KEY ON ERROR~GOTO TROFF TRON RESUME
	数値関数	ABS (X) ATN (X) CDBL (X) CINT (X) COS (X) CSNG (X) EXP (X) FIX (X) INT (X) LOG (X) RND (X) SGN (X) SIN (X) SQR (X) TAN (X)
	string	ASC (S) CHR\$ (X) HEX\$ (X) INKEY\$ INPUT\$ (X) INSTR ((X,) S1, S2) LEFT\$ (S, X) LEN (S) MID\$ (S, X, Y) OCT\$ (X) RIGHT\$ (S, X) SPACE\$ (X) STR\$ (X) STRING\$ (X, S又はY) VAL (S)
	その他の関数	CSRLIN DATE\$ ERL ERR FRE (X) INP (X) LPOS (X) PEEK (X) POINT (X, Y) POS (X) SPC (X) TAB (X) TIME\$ USRn (X) VARPTR

コマンドとステートメント

n, X, Y, Z は数値
S は文字列とする

デ ィ ス ク	コマンド	FILES FORMAT LFILES LOAD MERGE MOUNT NAME REMOVE SAVE SET
	ステートメント	FIELD LSET RSET
	入出力ステートメント	CLOSE GET# INPUT# KILL LINEINPUT# OPEN PRINT# PRINT#USING PUT#
	ストリング	INPUT# (X ([(n)]))
	ディスク関数	CVD (S) CVI (S) CVS (S) DSKI\$ (X, Y, Z) EOF (n) FPOS (n) LOC (n) LOF (n) MKD\$ (X) MKI\$ (X) MKS\$ (X) ATTR\$ (X) DSKF (n)
カ セ ッ ト テ ー プ	コマンド	CSAVE CLOAD CLOAD?
	入出力ステートメント	INPUT#-1 PRINT#-1
	特殊ステートメント	MOTOR

PC-8001のN-BASICはマイクロソフト社のM-BASICを土台にしている。さらに図形に関する命令などが追加されている。

大部分のマイクロコンピュータがM-BASICを土台にしているので、命令も機能も似かよったものが多い。

< >で囲まれたものは使用者が指定する。

[]で囲まれたものは必要なら指定する。

(ディスク)とあるのは、ディスクが接続されているときに使える。

(1) コマンド

AUTO [(<開始行>), (<間隔>)]

AUTO (RETURNキーを押す。以下同じ) で自動的に行番号を発生させる。停止はSTOPキーを押す。

AUTO.....10番から10の間隔で10, 20, 30と発生させる。

AUTO100, 100.....100, 200, 300.....

AUTO, 5.....10, 15, 20, 25.....

CLOAD " <ファイル名> "

カセットテープからプログラムをロードする。

CLOAD " ABC "ABCと名前のついたプログラムをロードする。

CLOAD " トリタテ "トリタテと名前のついたプログラムをロードする。

CLOAD? " <ファイル名> "

CSAVE後、正しくセーブ (記録) されたかどうか検証する。
メモリーの中身は消えない。

CONT

STOPで停止したプログラムの実行を再開する。ステートメントのSTOP参照。

CSAVE " <ファイル名> "

プログラムをカセットテープにセーブ (記録) する。

DELETE [(<開始行>) (-<終了行>)]

指定した行を抹消する。

DELETE 3030行を抹消。30と入力して RETキー
を押したのと同じ。

DELETE 100-150100行から150行を抹消。

DELETE -5050行までを抹消。

(ディスク) FILES [<ドライブ番号>]

ディスクに存在するファイル名と大きさを画面に一覧表示する。ドライブ番号を指定しなければ1番が指定される。

(ディスク) FORMAT [<ドライブ番号>]

ディスクをフォーマットして、使用可能にする。

第一レベルのフォーマット

FORMATステートメントを実行する。これでディスク(PC-8031)で読み書きできる。

BASICからもDSKI\$, DSKO\$で入出力ができる。他のコマンドやステートメントは使えない。

第二レベルのフォーマット

PC-8031付属のシステムディスクにあるformatプログラムを使う。FAT, ディレクトリー, IDの初期化ができる。これでBASICのコマンドやステートメントが実行できる。

KEYLIST

ファンクションキーのf・1からf・10の内容を画面に表示する。

(ディスク) LFILES

プリンターへFILESを出力する。

LIST [<開始行>] [- [<終了行>]]

プログラムを画面表示する。

LIST.....全部表示
LIST, -.....該当行からを表示。
LIST-,該当行までを表示。
LIST400.....400行のみを表示。
LIST400-.....400行からを表示。
LIST-400.....400行までを表示。
LIST100-400.....100行から400行を表示。
LIST,エラーになったときなどの該当行を表示。

LLIST [<開始行>] [- [<終了行>]]

プリンターへプログラムを出力する。使いかたはLISTと同じ。

(ディスク) LOAD " <ファイル名> "

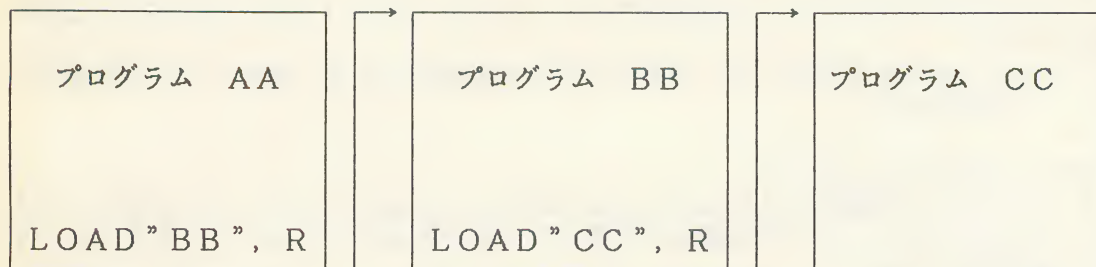
プログラムをディスクからメモリーへロード (転送) する。

下の手順でロードできる。

MOUNT 1
LOAD "トリタテ"
REMOVE

(ディスク) LOAD " <ファイル名> ", R

それまで開いていたデータファイルはそのまま、指定したプログラムをロードして実行する。大きなプログラムを分割して実行するときなどに使う。



BBがロードされると、AAもそこで使っていた変数も消える。データの受け渡しはディスクのデータファイルを仲介させる。

(ディスク) MERGE " <ファイル名> "

メモリー上にあるプログラムに対して、ディスクにあるファイルをまぜ合わせる。同じ行番号は、ディスクファイルのプログラムと置きかわる。ファイルはアスキー形式 (SAVE " <ファイル名> ", A) でセーブされたものであること。

MON

機械語モニターに制御を移す。アスタリスク (*) が出る。モニター・モードから BASIC・モードに戻るには、CTRLキーとBを押す。

モニター・モードで使えるコマンドは次のとおり。

- | | |
|---------------|---|
| * SXXXX | セット・メモリー。メモリーに機械語を書き込む。
XXXXは16進の番地。 |
| * DXXXX, YYYY | ダンプ・メモリー。XXXX番地からYYYY番地の内容を画面表示する。 |
| * L | ロード・テープ。テープから機械語を読み込む。終了すると*が出る。 |
| * WXXXX, YYYY | XXXX番地からYYYY番地の機械語をテープへセーブする。 |
| * LV | セーブした内容をベリファイする。 |
| * GXXXX | XXXX番地へジャンプする。機械語だけのプログラムの場合は、これで動き出す。 |
| * TM | メモリーのテスト。 |
| * CTRLキー + B | リターンベシック。モニターからBASICに戻る。 |

(ディスク) MOUNT [<ドライブ番号> [, <ドライブ番号>...]]

ディスクを設定する。MOUNTだけならすべてのドライブが対象となる。

(ディスク) NAME "<旧ファイル名>" AS "<新ファイル名>"

ファイル名を変更する。

NEW

メモリー上のプログラムと変数をクリアーする。

RENUM [[<新行番号>] [, <旧行番号>] [, <間隔>]]

行番号をつけ直す。GOTO文の行き先も自動的にかわる。

RENUM.....10, 20, 30.....とつけ直す。

RENUM100, 35, 20.....旧35行を100行として、以降20行間隔につける。

RENUM100, , 50.....100, 150, 200.....とつけ直す。

(ディスク) REMOVE [<ドライブ番号> [, <ドライブ番号>...]]

MOUNTしてあるディスクを抜きとる前に、FAT (58ページ) の書き込みをやり、抜きとってもいい状態にする。ライト・プロテクト・ノッチの銀紙がはりつけられてある場合には、REMOVEは不要である。

RUN [<行番号>]

プログラムを実行する。行番号を指定すると、そこから実行する。

(ディスク) RUN "<ファイル名>" [, R]

ディスクからファイルをロードし実行する。実行前に開いているファイルは閉じて、メモリーはクリアーする。Rをつけるとデータファイルは開いたまま。

(ディスク) SAVE "<ファイル名>" [, A]

下の手順でセーブできる。

プログラムをディスクへ保管する。Aをつけると、アスキー・コード (キャラクターコード表のコード) でプログラムをセーブする。つけなければ、バイナリー形式になり圧縮してセーブされる。REMOVEを絶対に忘れないこと。

MOUNT1
SAVE"トリタテ"
REMOVE

(ディスク) SET [<ドライブ番号または#ファイル番号または"ファイル名">, [<属性文字>]]

ファイルの属性をきめる。

属性文字は

R リード・アフターライト。書いたあと読む。
P ライトプロテクト。書き込み禁止。

SET 1, "R" 1 番ドライブへの出力は、R 検査をする。ただし、ディスクがマウントされている間だけ。

SET #1, "R" ファイル 1 が開かれている間、そこへの出力は R になる。

SET "ABC", "P" ファイル "ABC" を削除・変更から保護。

SET 1, "" ドライブ 1 のすべての属性をなくす。

TERM<コード>, <パリティ>, <ボーレート>, <フィールドスイッチ>

ターミナルモードにする。ターミナルモード参照。

コード	A	アスキーコード
	J	JISコード
パリティ	0	NON
	1	ODD
	2	EVEN
ボーレート	0	基準周波数×64
	1	基準周波数×16
フィールドスイッチ	0	OFF
	1	ON.....キャリッジリターンコードを受信すると自動改行。

(2) ステートメント

CLEAR [(<文字領域の大きさ>), <メモリーの上限>]

変数を 0, 文字変数をヌルストリング (" ") にクリアーして、文字領域の大きさ (初期値は 300) と BASIC が使えるメモリーの上限を指定。

CLEAR 500, &H DFFF 文字領域を 500 に、メモリー上限を &H DFFF 番地にする。次の &H E000 番地から、機械語を入れることができる。

DATA<定数> [, <定数>.....]

READ 文で読まれる数値や文字を割り当てる。READ 文と対で使う。

READ A, B, C
READ A\$, B\$, C\$

DATA 1, 5, 7, X, Y, Z

DEF FN x (<引数の並び>)

ユーザー関数を定義する。

```
10 DEF FNA (R) = 3.14 * R ^ 2
20 X = FNA (5)
30 Y = FNA (10)
40 PRINT X; Y
```

関数名をAとした。変数Rは10行のなかだけで対応していればよい。

DEFUSR [<数字>] = <番地>

機械語サブルーチンの開始番地を定める。数字は0から9まで。与えなければ0。

```
10 DEFUSR 2 = &HE000
.
.
80 Y =USR 2 (X)
```

BASICでXを入力すると、機械語サブルーチンが計算して、結果をYにもどす。BASICとサブルーチンの仲立ちはレジスターがする。ユーザールーチン参照。

DEFINT<変数名> [<-変数名>]
DEFSNG<変数名> [<-変数名>]
DEFDBL<変数名> [<-変数名>]
DEFSTR<変数名> [<-変数名>]

指定した変数の型を上から整数、単精度、倍精度、文字型に定義する。ただし、型宣言文字(%、!、#、\$)が、つねに定義に優先する。

```
DEFINT A-E
A, B, C, D, Eの文字で始まる変数を整数型に定義する。
DEFDBL F
文字Fで始まる変数を倍精度に定義する。
```

DIM<変数名> (<要素の数> [, <要素の数>.....])

配列の大きさを決めて、メモリーに領域を確保する。

DIM A (2)

A (0)	A (1)	A (2)
-------	-------	-------

 まで3個を確保。

DIM X (5)

X (0)	X (1)	X (5)
-------	-------	-------	-------

 まで6個を確保。

テキスト第6回、第7回の例を参照。

DIM Y (2, 3) 2次元で確保する。次のようになる。

	0	1	2	3
0	Y (0, 0)	Y (0, 1)	Y (0, 2)	Y (0, 3)
1	Y (1, 0)	Y (1, 1)	Y (1, 2)	Y (1, 3)
2	Y (2, 0)	Y (2, 1)	Y (2, 2)	Y (2, 3)

END

プログラムの実行を終了する。

ERASE<配列名> [, <配列名>.....]

配列を抹消する。したがって、そのメモリスペースを別用途に使える。
DIMA (x)などを抹消せずにDIMA (y)などと再宣言するとRedimensioned arrayエラーになる。

(ディスク) FIELD [井] <ファイル番号>, <フィールドの大きさ>AS<文字変数>.....

ランダムファイルバッファの中に変数用領域を割り当てる。ファイル番号は、そのファイルを開いたときに指定したもの。フィールドの大きさは、文字変数に割り当てるバイト数。文字変数がCで始まるときはASとの間にスペースを入れること。

FIELD1, 10ASA\$, 20ASB\$, 50ASC\$

ランダムバッファの最初の10バイトをA\$に、次の20バイトをB\$に、次の50バイトをC\$に割り当てる。合計で255バイトまで割り当てられる。FIELD文の変数名をINPUT文やLET文で使わないこと。使うと文字領域と混ざりあって、ランダムバッファの位置を正しく指さなくなる。

FOR<変数名>=<式>TO<式> [STEP<式>]

NEXTまでの処理を条件によってくりかえす。STEP<式>は、増加(減少)の幅。なければ1ずつ増加する。FOR~NEXTと対で使う。


```
10 FOR K=0 TO 10 STEP 2
20 PRINT K;
30 NEXT K
```

上のプログラムは実行すると、0 2 4 6 8 10と結果が画面表示される。
また、10 FOR K =N TO 10 として、N>10の場合でも1回は実行する。


```

FOR A=....
FOR B=....
.
NEXT B
NEXT A

```



多重ループの場合、内側のNEXT文は、外側のNEXT文の前にあること。

GOSUB<行番号>

指定した行からはじまるサブルーチンへとぶ。サブルーチンの最終行にはRETURNがあること。

GOTO<行番号>

指定した行へプログラムの制御を移す。

IF<条件式>THEN<文または行番号>〔ELSE<文または行番号>〕

条件式が真ならTHENのあとの命令を、偽ならELSEのあとの命令を実行する。THENとELSEの関係は、ELSEからみた左側のTHENで他のどのELSEにも対応していないTHENとの組になる。

LET<変数名>=<式>

変数に値を代入する。LETは省略できる。

```
LET A=3.14159*R^2
```

(ディスク) LSET<文字変数>=<文字表記>

PUT文の準備として文字領域をメモリーからランダムバッファに左づめで移す。数値は、ランダムバッファに移す前に文字にしなくてはならない。LSET (RSET)を実行する前に、FIELD文があること。

```

LSET A$="HAPPY"
LSET B$=MK$ (B)
LSET C$=X$

```

A\$, B\$, C\$はFIELD文で定義されていること。A\$が10バイトとてあれば、HAPPYと入る。

NEXT〔<ループ変数>〔,<ループ変数>〕〕

FORループの範囲を決める。FOR参照のこと。

ON<式>GOSUB<行番号>〔, <行番号>……〕

式の値に対応して、サブルーチンへとぶ。

```
ON A-1 GOSUB 100, 200, 500
```

A-1の値が1なら100行, 2なら200行, 3なら500行へ。その他の値だと次の行へ制御がうつる。ただし, 条件の式の値が負だとエラー。

ON<式>GOTO<行番号>〔, <行番号>……〕

式の値に対応した行番号へとぶ。

READ<変数>〔, <変数>……〕

DATA文のデータを各変数に読みこむ。DATA参照。

REM〔<任意の文字列>〕

プログラム中に注釈をつける。アポストロフィー（'）でもおなじ。マルチステートメントにはできない。

```
10 REM **KEISAN**
20 REM.....空白の1行ができる。
50 ' ** シ ユンヒ **
60 PRINT A ' ケッカノフ リント
```

RESTORE〔<行番号>〕

DATA文はREADで1回読むと, あとは使えない。それを解除して, DATA文をはじめてから読めるようにする。あるいは, 指定した行のDATA文から読めるようにする。したがって, 解除して読む場合, READ文の前にはRESTORE文が必ずあるはず。

RETURN

サブルーチンから, そのサブルーチン呼びだしたGOSUB文の次の行へもどる。したがって, サブルーチンの最後の文はRETURN。

```
50 GOSUB 200
60 PRINT A
.
200 A=25*X
300 RETURN
```

〕 200行, 300行がサブルーチン。
最後にRETURNがある。

(ディスク) RSET<文字変数>=<文字表記>

LSET参照。右づめで移す。

STOP

プログラムの実行を中止してブレークメッセージを画面表示、コマンドモードへもどる。END文とちがって、使用中のファイルを閉じないので、デバック中などにダイレクトモードで中間結果を調べたり、変更したりできる。CONTまたはGOTO行番号で再開できる。プログラムを変えるとCONTはきかない。

SWAP<変数>, <変数>

ふたつの変数の値を交換する。変数の型は一致していること。

```
10 A=3, B=4
20 SWAP A, B
30 PRINT A, B.....A=4, B=3になる。
```

(3) 入出力ステートメント

(ディスク) CLOSE [(#) <ファイル番号> [, (#) <ファイル番号>]]

ディスクファイルへの入出力を終了する。
CLOSEだけだと、オープン状態のすべてのファイルをクローズする。END文とNEWコマンドでもそうなる。STOPでは、ディスクファイルのクローズはやらない。

(ディスク) DSKO\$<ドライブ番号>, <トラック番号>, <セクター番号>

ドライブ, トラック, セクターを指定して, そのセクターにFIELD#0で指定したランダムバッファの内容を書きこむ。MOUNTコマンドなしで動いてしまう。したがって, ファイル・アロケーション・テーブルやディレクトリーと無関係に動くので危険。DSKO\$はステートメントだが, 逆の機能を持つディスク関数にDSKI\$がある。

(ディスク) GET#<ファイル番号> [, <レコード番号>]

ランダムディスクファイルからランダムバッファに1レコード読みこみ、同時にFIELD文で指定した各フィールドに対応した変数にそのデータを代入する。レコード番号は最大544まで。省略すると最後にGETした次のレコードが、バッファに読みこまれる。

INPUT (" <プロンプト文> " ;) <変数> [, <変数> ……]

キーボードから入力する。変数は数値変数名でも添字つき変数名でも文字変数でもよい。文字列の入力は、引用符で囲む必要はない。

INPUT " ツギノデータ " ; A

(ディスク) INPUT # <ファイル番号> , <変数名> [, <変数名> ……]

ディスクのシーケンシャルデータファイルからデータを読み、プログラムの変数に代入する。ファイル番号が-1のときは、カセットからの入力。

(ディスク) KILL " <ファイル名> "

ディスク上から指定したファイルを削除。

LINE INPUT (" <プロンプト文> " ,) <文字変数>

特殊文字を含めて1行全体(255文字以内)を区切らずに入力する。

(ディスク) LINE INPUT # <ファイル番号> , <文字変数>

1行全体を区切らずにシーケンシャルデータファイルから文字変数に読む。

LPRINT <式> [, または ; <式> ……]

プリンターへ出力。PRINT参照。

LPRINT USING " <フォーマット文> " ; <式> [, <式> ……]

プリンターへ出力。PRINT USING参照。

(ディスク) OPEN " <ファイル名> " [FOR <モード>] AS [#] <ファイル番号>

データの入出力対象となるディスクファイルの使用宣言。
モード INPUT …… 入力ファイルの指定。ファイルポインターは、そのファイルの先頭を指す。ファイルがないと

エラーになる。

OUTPUT.....出力を行う。ファイルがあれば、その先頭をファイルポインターは指す。ファイルがなければ新しいファイルを作り、ファイルポインターはその先頭を指す。

APPEND.....追加。そのファイルの最後のレコードの直後をファイルポインターは指す。

省略すると.....ファイルがあれば、その先頭をファイルポインターは指す。なければ、そのファイル名のファイルを作り、ファイルポインターはその先頭を指す。

OUT<ポート番号>, <データ>

出力ポート (0~255の整数) にデータ (0~255の整数) を送る。

OUT &H40, &H20.....BEEP1
OUT &H40, &H00.....BEEP0

POKE<メモリー番地>, <データ>

メモリー内の指定した番地にデータを書き込む。メモリー番地は、&H0000 (10進数の0) から&HFFFF (10進数の65535) の範囲、データは&H00 (0) から&HFF (255) までの範囲であること。

PRINT<式> [, または ; <式>.....]

画面表示する式は数値や文字でもかまわない。式を省略すると改行される。

コンマ (,) で区切ると、14字ずつ区切られる。セミコロン (;) で区切ると、次の値は連続してプリントされる。数値のプリントは、頭に符号の入るスペース、末尾にスペース1個がとられる。文字列は引用符 (") で囲むこと。式の最後にコンマもセミコロンもなければ、改行される。キーワードPRINTの代りに疑問符 (?) が使える。LPRINTの代りにL?は認められない。引用符 (") とセミコロン (;) が隣接するときには、セミコロンを省略できる。

PRINT USING<"フォーマット文">;<式のリスト>

文字や数値を指定したフォーマットでプリントする。PRINT USINGの一覧表を参照のこと。

(ディスク) PRINT#<ファイル番号>, (USING<フォーマット文>;)
)<式のリスト>

シーケンシャルファイルにデータを書く。ファイル番号は、ファイルがオープンされたとき指定されたもの。-1だとカセットテープへの出力になる。フォーマット文はPRINTUSINGのフォーマットと同じ。式のリストを複数書くときにはセミコロン

(;)で区切ること。カンマ(,)で区切ると余分な空白も書きこむ。

```
A$="TOKYOTO"  
B$="CHIYODAKU"  
PRINT#1,A$;B$
```

はTOKYOTOCHIYODAKUとディスクに書かれる。

```
PRINT#1,A$;" ";B$
```

とするとTOKYOTO, CHIYODAKUとなり、2つの文字変数としてふたたび読める。

(ディスク) PUT#<ファイル番号> [, <レコード番号>]

ランダムディスクファイルにランダムバッファから1レコードを書き込む。レコード番号が省略されると、PUT文で最後に書いた次の番号が割り当てられる。レコード番号の上限は544。

WAIT<ポート番号>, <マスク>, <論理スイッチ>

入力ポートをモニターする。(ポートから読みこんだデータ XOR 論理スイッチ) AND マスク が真ならプログラムの実行を再開する。32~33ページ参照。

(4) スクリーンステートメント

COLOR, LINE, PSETなどでつかわれるファンクションコードのアトリビュート(属性)

コード	白黒モード	カラーモード
0	ノーマル	黒
1	シークレット	青
2	ブリンク	赤
3	シークレット	紫(マゼンダ)
4	リバーズ	緑
5	リバーズシークレット	水色(シアン)
6	リバーズブリンク	黄
7	リバーズシークレット	白

COLOR [<ファンクションコード>] [, <ヌルキャラクターコード>] [, <グラフィックスイッチ>]

画面に対する色や機能を指定する。

ヌルキャラクターコード.....画面をクリアしたときに表示されるコード。アスキーコードか文字で指定する。通常は0。

グラフィックスイッチ.....1 グラフィックモード
0 キャラクターモード

CONSOLE [<スクロール開始行>] [, <スクロールの長さ>] [, <ファンクションキーの表示>] [, <カラーか白黒か>]

スクロールとは画面をせりあがらせること。指定した範囲がスクロールの対象になる。対象外の画面はそのままですりあがらない。

ファンクションキー.....1 画面下のファンクションキーを表示。
0 画面下のファンクションキーを表示しない。
カラー指定.....1 カラーモード
0 白黒モード

GET@ (A) (X1, Y1) — (X2, Y2), <配列名> [, G]

画面上の情報を指定した配列に読み込む。PUT@と一対で使われる。PUT@を参照のこと。

GET@ならPUT@で受ける。

GET@A (アトリビュートつき。色やリバースなどを同時にGETする) ならPUT@Aで受ける。

GET@.....G (グラフィックモード) ならPUT@.....<条件>で受ける。

LINE<画面の行>, <ファンクションコード>

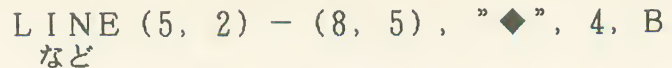
カラーモードの場合1行単位で、ファンクションを指定。
色はCOLORで指定する。

ファンクションコード.....0 ノーマル
1 ブリンク
2 リバース
3 リバースブリンク

白黒モードでは使えない。Syntax Errorになる。

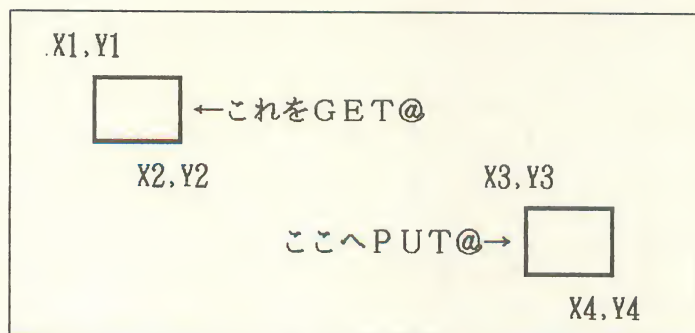
LINE (X1, Y1) — (X2, Y2), "<文字列>" [, B [F]]

画面に文字を使って、線や箱を描く。色指定もできる。
Bをつけると箱が描け、BFとすると箱のなかもうまる。



PUT@ (A) (X3, Y3) - (X4, Y4), <配列名> [, <条件>]

GET@で作成した配列を画面の任意の部分に出力する。



GET@とPUT@の面積が同じなら、形がかわっても移せる。

```
10 LOCATE 10, 5:PRINT"ABCDE"
20 LOCATE 0, 0:DIM P%(2)
30 GET@(10, 5) - (14, 5), P%
40 PUT@(20, 5) - (20, 9), P%
```

	10	11	12	13	14	15	16	17	18	19	20
5		A	B	C	D	E					A
6											B
7											C
8											D
9											E

上の5つの文字が →

ここへ複写される

(キャラクターモード) GET@ならPUT@で受ける。

(Aつき) GET@AならPUT@A

アトリビュート (属性) つき。色やリバースなども同時に移せる。

(グラフィックモード) GET@...GならPUT@...<条件>

GつきGET@は、画面にドットがあれば1, なければ0と記憶していく。PUT@文の末尾の条件で次表のようになる。1はドットがある。0はない。

条 件	GET@のドット	1		0	
	PUT@する画面の状態	1	0	1	0
	PSET	1	1	0	0
	PRESET	0	0	1	0
	AND	1	0	0	0
	OR	1	1	1	0
	XOR	0	1	1	0
	NOT	0	0	1	1

GET@ (0, 0) - (15, 15), A%, G
 PUT@ (25, 40) - (40, 55), A%, OR

配列の大きさの計算のしかた

それぞれ割った数を切り上げて使う。

(1)キャラクターモード……1文字しまいこむには1バイト(8ビット)必要。

必要なバイト数 / b

(2)Aつきのとき

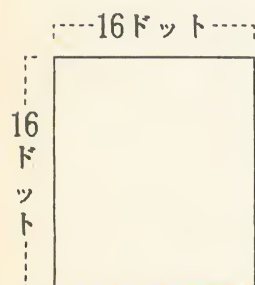
必要なバイト数 × 2 / b

(3)グラフィックモード

((必要なドット数) / 8 + 2) / b

bは——

- 整数型配列 (例P%) のとき 2
- 単精度型配列 (例A) のとき 4
- 倍精度型配列 (例A#) のとき 8



左のグラフィックモードを整数型配列P%(I)でとるには、
 $I = (16 * 16 / 8 + 2) / 2$

したがってP%(16)。配列はP%(0)からある。したがってP%(16)で17ある。

WIDTH [<水平の桁数>] [, <垂直の行数>]

画面に表示する文字の桁数、行数を指定する。組み合わせは下のとおり。

桁数 36, 40, 72, 80
 行数 20, 25

(5) 特殊ステートメント

BEEP [<1または0>]

内蔵ブザーから音を出す。

BEEP.....一定時間(約0.4秒)鳴ってとまる。

BEEP 1.....連続して鳴る。BEEP 0(ゼロ)で停止する。

ERROR [<エラーコード>]

(1)エラー発生をシミュレートする。ON ERROR GOTOと組みにして使う。

(2)BASICで使っていないエラーコードを定義して、エラー処理ルーチンで使う。

コードは0~255だが、BASICでエラーコードとして定義されているものと重複しないように注意する。

30 IF A>1000 THEN ERROR 200.....定義する。

80 IF ERROR=200 THEN ~使う。

KEY<キー番号>, "<文字列>"

f・1からf・10のファンクションキーを定義する。最大15文字まで文字列とコントロール文字を指定できる。

KEY 1, "WIDTH 40, 20" + CHR\$(13)

KEY 2, "REM*****"

MOTOR [<スイッチ>]

カセットテープレコーダーのモーターを制御する。

スイッチ.....0.....モーターがとまる(off)

0以外の数.....モーターが動く(on)

スイッチの指定がなければonをoffに、offをonにする。

ON ERROR GOTO<行番号>

エラー発生時のとび先を定義する。

TROFF

トレースモードを解除する。

TRON

トレースモードに入り、プログラムの実行状態を追跡する。実行したプログラムの行番号が次つぎにカギカッコで囲まれて画面表示される。

```
10 TRON
20 FOR K=5 TO 7
30 PRINT K
40 NEXT K
```

RUN

(20) (30) 5.....ループ1回目, 5がPRINTされる。

(40) (30) 6.....ループ2回目, 6がPRINTされる。

(40) (30) 7.....ループ3回目, 7がPRINTされる。

(40)

10行のTRONが実行されると、以後はトレースモード。TROFFで切る。

RESUME (<0またはNEXTまたは行番号>)

エラーに対する処理をすませたあと、RESUMEでプログラムの実行を再開する。

RESUME 0.....エラー行からプログラム再開する。

RESUME NEXT.....エラー行の次の行から再開する。

RESUME <行番号>.....指定の行から再開する。

ON ERROR GOTO <行番号>で、エラー処理ルーチンへとぶ。エラー処理したら、プログラムを終了させるのか再実行させるのかの判断が必要になる。したがってエラー処理ルーチンの最後には、ENDかRESUMEが要る。

(例) WAIT

キーボードの入力ポートは、キーが押されないかぎり255 (&HFF) の状態である。すなわち11111111とビットがすべて立つ。WAITも入力ポートのデータを読むが、データがある状態になるまで待つ。キーボードマップ参照のこと。

WAIT P, J, K

P....ポート番号。 J....データ

K....セットデータ。セットしなければ0とみなす。

まず、P XOR K を実行する。結果をQとすれば、Q AND J を実行して、0であれば再度ポートの状態を読んで同じことを繰り返す。0でなければ次のプログラムへ進む。

たとえば、テンキーの「3」が押されたら、ABCDEFGHとプリントするプログラムは次のようになる。


```

10 WAIT 0, 8, 255
20 PRINT "ABCDEFG"

```

なにもしないPの状態 = 255 = 1 1 1 1 1 1 1 1
 K = 255 = 1 1 1 1 1 1 1 1

(P XOR K) = Q = 0 0 0 0 0 0 0 0
 J = 8 = 0 0 0 0 1 0 0 0

(Q AND J) = 0 0 0 0 0 0 0 0

3のキーを押したPの状態 = 247 = 1 1 1 1 0 1 1 1
 K = 255 = 1 1 1 1 1 1 1 1

(P XOR K) = Q = 0 0 0 0 1 0 0 0
 J = 8 = 0 0 0 0 1 0 0 0

(Q AND J) = 0 0 0 0 1 0 0 0 = 8

結果が0でないから、次の20行が実行されてABCDEFGがプリントされる。
 WAITは使いかたによっては無限ループに入ってしまう。30 GOTO 10
 を追加すると、STOPキーではとまらない。STOPキーを押しながらリセットキ
 ーを押せばプログラムは助かる。

一般的な使いかたは、たとえばプリンターがBUSY（稼働中）かREADY（待機
 中）かをWAITを使ってステータスビットを調べて判断し、OUTでデータを送り
 だす。さもないと、ほかの仕事で稼働中のところへさらにOUTでデータを送ってし
 まい、データを消滅させてしまう。

(例) INP

Kのキーをおすとベルがなる。

```

10 IF INP (3) = 247 THEN 30
20 GOTO 10
30 BEEP1 : FOR J = 0 TO 50 : NEXT : BEEP0 : GOTO 10

```

INP (x) は、その他の関数にある通りポート番号x番から入力したデータを与え
 る。わかりやすく、キーボードで説明してみる。

キーボードマップをみると、縦に00から09までのインプットアドレスがあって、
 横にD0からD7までのデータバスがある。コンピューターが稼働中なら、常時00
 から09のインプットアドレスを監視して、キーが押されたかどうか見張っている。
 キーがなにも押されない状態では、D0からD7はすべてビットが立っている（負論
 理という。255すなわち&HFFの状態）。キーが押されるとそこだけ1が0にな
 る。

	1	2	4	8	16	32	64	128	
	D0	D1	D2	D3	D4	D5	D6	D7 ← D7が最上位ビット	
キーが押されていない	1	1	1	1	1	1	1	1	= 255 = &HFF
「1」が押された	1	0	1	1	1	1	1	1	= 253 = &HBF
「K」が押された	1	1	1	0	1	1	1	1	= 247 = &HEF

数値関数

x は数値を示す

関 数	機 能	備 考
ABS (x)	x の絶対値 $ x $ を与える	PRINT ABS (-40) は 40 PRINT ABS (A+8) など
ATN (x)	x のアーктanジェント	x はラジアン。ラジアンと変数の 関係は $1 \text{ ラジアン} = 180^\circ / \pi$ $\therefore 1^\circ = 3.14159 / 180$
CDBL (x)	x を倍精度数値に変換	A = CDBL (x) など
CINT (x)	x を整数に変換	PRINT CINT (34.56) は 34 $-32768 \leq x \leq 32767$ のこと
COS (x)	x のコサイン	x はラジアン $1 \text{ ラジアン} = 180^\circ / \pi$
CSNG (x)	x を単精度変数に変換	A = CSNG (x+2) など
EXP (x)	e の x 乗	PRINT EXP (1.60944) は 5.00001 $e = 2.71828459$ LOG との関係は $y = \log_e x$ $x = e^y$
FIX (x)	小数点以下を切りすてた 整数	PRINT FIX (4.56) は 4 PRINT FIX (-4.5) は -4
INT (x)	x を超えない最大の整数	PRINT INT (4.56) は 4 PRINT INT (-4.56) は -5
LOG (x)	x の自然対数	$x > 0$ のこと。自然対数を常用対 数にする式は $\text{LOG} (x) / \text{LOG} (10)$

数値関数

x は数値を示す

関数	機能	備考
RND (x)	乱数をつくる	x はダミー。結果は $0 < \text{乱数} < 1$ になる。 A=RND (1) など x > 0 なら通常の乱数 x = 0 なら同系列乱数 x < 0 なら乱数系列を更新する
SGN (x)	x の正負符号を与える	A=SGN (x) x > 0 なら A=1 x = 0 なら A=0 x < 0 なら A=-1
SIN (x)	x のサイン	x はラジアン 1 ラジアン = $180^\circ / \pi$
SQR (x)	x の平方根	A=SQR (2) など。x ≥ 0 であること
TAN (x)	x のタンジェント	x はラジアン 1 ラジアン = $180^\circ / \pi$

ストリング関数

x, y は数値, S は文字列とする

関 数	機 能	備 考
ASC (S)	文字列Sの最初のコードを数値で与える	A\$ = "ABC" PRINT ASC (A\$) は 65 BとCは無視される
CHR\$ (x)	コードxの対応する文字や機能を与える ASCの逆	PRINT CHR\$ (7) は BEEP 1 PRINT CHR\$ (84) はT PRINT CHR\$ (&HF1) は円
HEX\$ (x)	数値xの16進数文字列を与える	PRINT HEX\$ (80) は 50
INKEY\$	キーボードが押されたかどうか調べる。キーボードが押されればその文字を与える	10 X\$ = INKEY\$ 20 IF X\$ = "" THEN 10
(ディスク) INPUT\$ (x [, (#) y])	キーボードまたは番号yのファイルから入力したx文字分の文字列	キーボードから入力された文字は画面表示されない。すべてのコントロール文字はコントロールCをのぞき、そのまま入力される X\$ = INPUT\$ (2, #1) など
INSTR ((x,) S1, S2)	S1からS2と同じ文字列を探し、合致したら最初の文字がS1の何文字目かを与える 同じ文字列が見つからなければ0	X\$ = "ABCABCD" : Y\$ = "CD" INSTR (X\$, "C") は3 INSTR (4, X\$, Y\$) は6 xを与えるとS1のx文字目から探しはじめる
LEFT\$ (S, x)	Sの左からx番目までの文字列を与える	X\$ = "ABCDEF" LEFT\$ (X\$, 3) はABC
LEN (S)	Sが何文字あるかを与える	X\$ = "ABCDEF" PRINT LEN (X\$) は6
MID\$ (S, x, y)	Sの中の左からx文字目から長さyの文字列を与える 次ページへ続く	X\$ = "ABCDEF" Y\$ = MID\$ (X\$, 3, 2) : PRINT Y\$ はCD

ストリング関数

x, y は数値, S は文字列とする

関数	機能	備考
MID\$ つづき	MID\$ は式の左辺でも使うことができる	<pre> 10 A\$ = "AAAAA" 20 MID\$ (A\$, 2, 2) = "●●" 30 PRINT A\$ 40 END </pre> <p>上例では20行でMID\$ が式の左辺に使われている プログラムを実行すると、結果は次のようになる A●●AA</p>
OCT\$ (x)	数値 x の 8 進数文字列を与える	PRINT OCT\$ (&H11) は 21
RIGHT\$ (S, x)	S の右から x 番目までの文字列を与える	<pre> X\$ = "ABCDEF" PRINT RIGHT\$ (X\$, 3) </pre> <p>は DEF</p>
SPACE\$ (x)	x 個の空白を与える	<pre> X\$ = "ABC" : Y\$ = "DEF" C\$ = X\$ + SPACE\$ (7) + Y\$ PRINT C\$ </pre> <p>は ABC←スペース7個→DEF</p>
STR\$ (x)	数値 x をそれを表示する文字列に変換	<pre> PRINT STR\$ (1234) + "円" </pre> <p>は 1234円</p>
STRING\$ (x, <S または y >)	S の最初の文字を x 個または、アスキーコード y の文字を x 個与える	<pre> PRINT STRING\$ (5, "ABC") </pre> <p>は AAAAA</p> <pre> PRINT STRING\$ (5, &HEC) </pre> <p>は ●●●●●</p>
VAL (S)	数字を数値へ変換する	<p>S の最初の文字が +, -, &, 数字でなければ 0 になる</p> <pre> PRINT VAL ("010") </pre> <p>は 10</p> <pre> PRINT VAL ("A10") </pre> <p>は 0</p>

ディスク関数

x は数値, S は文字列とする

関 数	機 能	備 考
CVD (S)	8 バイトの文字列を倍精度の数値へ変換する	ランダムファイルから読み出した数値は、文字列から数値に変換する必要がある 10 FIELD#1, 4AS A\$, 20AS B\$.... 20 GET#1 30 A=CVS (S)
CVS (S)	4 バイトの文字列を単精度の数値へ変換する	
CVI (S)	2 バイトの文字列を整数の数値へ変換する	
DSKI\$ (<ドライブ番号>, <トラック番号>, <セクター番号>)	指定したセクターの内容を変数名と#0のフィールドバッファに与える	A\$=DSKI\$ (1, 3, 5) DSKO\$ステートメントと逆の働きをする
EOF (<ファイル番号>)	指定した番号のシーケンシャルファイルの終わりを示す	終りになると-1 (真) の値をとる。シーケンシャルファイルからの入力時, " INPUT PAST END " エラーをふせぐために使う
FPOS (<ファイル番号>)	ファイルのあるセクター番号	セクター番号が与えられることを除くとLOCと同じ
LOC (<ファイル番号>)	ランダムファイルの場合レコード番号を指定しないGETまたはPUTが実行されたときに、使用されたレコード番号が与えられる。シーケンシャルファイルの場合、ファイルが開かれてから読み出された (または書き込まれた) セクター数が与えられる	20 PRINT LOC (1)

ディスク関数

x は数値, S は文字列とする

関 数	機 能	備 考
LOF (<ファイル 番号>)	ランダムファイルのPUT, GETによりアクセスされた最大のレコード番号を与えられる	500 IF NO>LOF (1) THEN PRINT "ファイル END"
MKD\$ (x)	倍精度の数値 x を 8 バイトの文字列へ変換する	LSET, またはRSETによりランダムバッファに入れる数値はすべて文字列に変換しなければならない 10 OPEN"FILE"AS #1 20 FIELD#1, 20AS N\$, 4 AS A\$ 30 X\$="ABC":LSE T N\$=X\$ 40 A=34.56 50 RSET A\$=MKS\$ (A)
MKS\$ (x)	単精度の数値 x を 4 バイトの文字列へ変換する	
MKI\$ (x)	整数 x を 2 バイトの文字列に変換する	
ATTR\$ (<ドライブ番号> または<ファイル番 号>または<ファイ ル名>)	指定したドライブやファイルの現在の属性を示す	属性を返す。 R....READ AFTER WRITE P....WRITE PROTEC T A\$=ATTR\$ (1)
DSKF (<ドライ ブ番号>)	指定したドライブにマウントされているディスクの未使用クラスタ数を与える	PRINT DSKF (1)

その他の関数

x は数値, S は文字列とする

関数	機能	備考
CSRLIN	画面上のカーソルの縦の位置を与える	LOCATE 4, 5:PRINT CSRLIN は 5 と出る POS 参照のこと
DATE\$	内蔵クロックの日付	PRINT DATE\$ 日付をセットするには, DATE\$ = "YY/MM/DD" で入る DATE\$ = "81/04/12" などを入れる
ERL	エラーが起きた行番号を与える	IF ERL = 130 THEN 50
ERR	エラーの番号を与える	IF ERR = 4 THEN 100
FRE (x)	ユーザー用メモリーエリアの空きを示す。x はダミー	PRINT FRE (0) 電源投入後, RAM16K のときは, 10402, 32K ならば 26786
INP (x)	I/O アドレス x のポートから読んだデータの値	A = INP (2) 33 ページに例がある。キーボードマップとあわせて参照のこと
LPOS (x)	プリンターのヘッドの位置。x はダミー	1 行 60 文字で改行したければ, 最大 80 文字の場合 IF 80 - LPOS (1) = 20 THEN LPRINT
PEEK (x)	メモリー x 番地の内容を与える	A = PEEK (&H1056) 出力ステートメント POKE の逆
POINT (<水平位置>, <垂直位置>)	指定した位置にドットがあるかどうかを判断する	あれば -1 (真), なければ 0 (偽) となる 50 IF POINT (30, 40) = 0 THEN RETURN

その他の関数

x は数値, S は文字列とする

関数	機能	備考
POS (x)	画面上のカーソルの横の位置を与える	PRINT POS (0), CS RLIN これでカーソルの水平, 垂直位置がわかる
SPC (x)	プリント文中で使い, x 個の空白を与える	PRINT "A"; SPC (3)); "B" A△△△B (△は空白とする)
TAB (x)	プリント文中で使い, 左端から x 字まで空白とする。現在のカーソル位置 > x ならば無効	PRINT TAB (5); "A" △△△△△A
TIME \$	内蔵クロックの時刻	時刻をセットするには TIME \$ = "HH:MM:SS" " TIME \$ = "23:00:15" "
USR<n> (x)	ユーザーが定義した n 番目の機械語ルーチンへとぶ	n は 0 から 9 まで。省略すると 0 x は引数 ステートメント DEFUSR 参照
VARPTR (<変数名>) VARPTR (#<ファイル番号>)	変数の割り当てられているアドレスを与える。ファイル番号を指定した場合には, 指定したファイル番号のディスクファイルに与えられた入出力バッファの開始アドレスを与える	アドレスは 32767 から -32768 まで。負の場合には, 65536 を加えると実数値になる

PRINT USING 一覧表

左 づ め で 表 示	!	与えられた文字列の最初の1文字だけをプリントする 10 A\$="ABC";B\$="DEF" 20 PRINT USING "!" ;A\$;B\$ AD
	& &	文字データの桁数を指定。空白+2が桁数 10 A\$="ABC";B\$="DEF" 20 PRINT USING "& &" ;A\$ ABC
右 づ め で 表 示	#	数字の桁数を指定 10 A=123 20 PRINT USING "####" ;A 123 30 PRINT USING "タイジ ユウ##. #Kg" ;56.7 タイジ ユウ 56.7Kg
	.	#と#の間へ、(ピリオド)をうつと、そこが小数点 10 PRINT USING "####.###" ;A 123.456 20 PRINT USING "####.##" ;A 123.45
	+	正の符号をつける。数値が負のときは負の符号がつく PRINT USING "+##.##" ;77.8 +77.8 PRINT USING "##.##+" ;-12.34 12.34-
	-	負の符号をつける。数値が正のときは正の符号はつかない PRINT USING "##.##-" ;-68.95 68.95- PRINT USING "##.##" ;-5.67 -5.67
	**	頭部の余白を*でうめる PRINT USING "***##.##" ;12.3 ***12.30 PRINT USING "***##.##円" ;2.54 ****2.5円
	¥¥	数値の頭に¥をつける PRINT USING "¥¥####" ;2000 ¥2000

PRINT USING 一覧表

右 づ め で 表 示	** π	前の2つの合成 PRINT USING " π #####"; 1234 *** π 1234
	,	整数部分を3桁ごとに, (コンマ) で区切る PRINT USING "#####", "; 1234 56789 123, 456, 789
	^ ^ ^ ^	指数型表示にする 10 A=123 20 PRINT USING "##.##^ ^ ^ ^"; A 1.23E+02 30 PRINT USING "##.##^ ^ ^ ^"; -0. 000005 -5.00E-06
%	数字の桁(井)が足りないと頭に%が表示される 数値を丸めた結果桁上りして桁(井)が不足すると%が表示される PRINT USING ".##"; 0.993 %1.00	

LPRINT USING とするとプリンターへ出力する。

<例>

```

10 A$="ASAHISHIMBUN"
20 B$="KS"
30 C=1000
40 LPRINT USING "&←空白10個→&  && #####円";
    A$, B$, C

```

結果

ASAHISHIMBUN KS 1000円

浮動小数点形式による数の表示

N-BASICでは、実数（1. 5や3. 14159など）はすべて浮動小数点形式で扱っている。

これは、365なら次のような表現になる。

$$3.65 \times 10^2$$

3.65を仮数、2を指数と呼ぶ。この方法を2進数でやると次のようになる。

10進

2進

0.5	1.0	×	2 ⁻¹
1.0	1.0	×	2 ⁰
1.25	1.01	×	2 ⁰
1.5	1.1	×	2 ⁰
2.0	1.0	×	2 ¹
0.0	0.0	×	2 ⁰

N-BASICの内部表現では、次のようになる。

*単精度 4バイトで表現する

下位8ビット	中位8ビット	上位8ビット	指数8ビット
----- 仮 数 -----			

*倍精度 8バイトで表現する

最下位8ビット	-----	最上位8ビット	指数8ビット
----- 仮 数 -----			

<指数>

指数は正負を表すため、実際の数値に129 (&H81)を足した値を使っている。

指数の値	+126	+1	0	-1	-128
内部表現	255	130	129	128	1

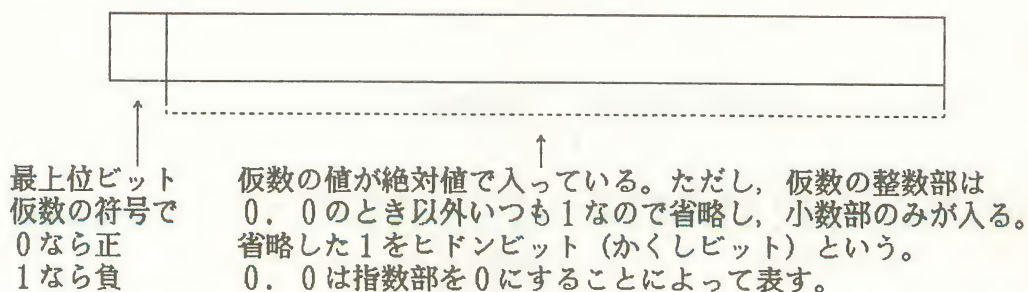
$$1 = 1.0 \times 2^{129-129} = 1.0 \times 2^0 = 1.0 \times 1$$

$$2 = 1.0 \times 2^{130-129} = 1.0 \times 2^1 = 1.0 \times 2$$

$$0.5 = 1.0 \times 2^{128-129} = 1.0 \times 2^{-1} = 1.0 \times 0.5$$

< 仮数 >

仮数は、正規化（桁あわせ）という操作によって上位ビットにつめられている。
仮数部を上位バイトから下位バイトへとならべると下図のようになる。



2進数で表した仮数

仮数部の内部表現

1. 0 0 0 -----

0 0 0 0 -----

1. 0 1 0 -----

0 0 1 0 -----

1. 1 0 0 -----

0 1 0 0 -----

-1. 0 0 0 -----

1 0 0 0 -----

↑
仮数の整数部
仮数の小数部

↑
仮数部の符号
仮数の小数部

以上の方法で -0.75 を内部表現になおすと次のようになる。

$$-0.75 = -0.11 \times 2^0 = -1.1 \times 2^{-1} = -1.1 \times 2^{128-129}$$

-----正規化-----

* 内部表現を画面に表示する方法

```

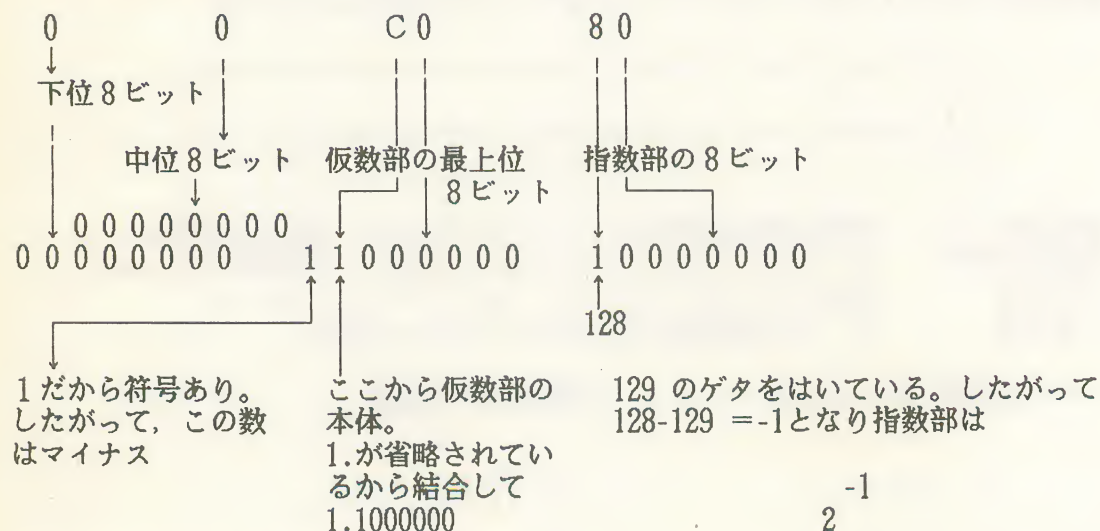
100 A=-0.75
200 FOR J=VARPTR (A) TO VARPTR (A) + n
300 PRINT HEX$ (PEEK (J)) ; " ";
400 NEXT J

```

単精度なら 3
倍精度なら 7

VARPTRは変数Aがしまわれている最初の番地を返すので、その番地から必要バイト数をPEEKで取り出して表示する。

実行結果は次のようになる。



以上をまとめると次のようになる。

$$-1.100000000000000000000000 \times 2^{-1} = -0.11$$

2進法の掛算だから桁がひとつずれて -0.11 になる

(参考)

$$2^{-1} = 1/2 = 0.5$$

$$2^{-2} = 1/4 = 0.25$$

$$2^{-3} = 1/8 = 0.125$$

$$2^{-4} = 1 / 16 = 0.0625$$

USR関数

N-BASICでは、機械語サブルーチンをUSR関数（ユーザー関数）として使うことができる。機械語サブルーチンを使う手順は次のようにする。

- (1) CLEAR文でサブルーチンを入れる場所を確保する。
通常BASICで使うアドレスの上限は59903 (&HE9FF) 番地だから、ここから機械語で必要とするスペース分だけ引いてセットしてやればよい。200語分必要なら次のようになる。(CLEAR文参照)

CLEAR 300, 59903-200

- (2) 確保した場所に、POKE文やモニター (MON) で機械語を書きこむ。
- (3) 機械語サブルーチンの開始番地をDEFUSR文で定義する。

DEFUSR n = 開始番地 (nは0から9の数, 与えなければ0)

- (4) 機械語サブルーチンを呼び出して使う。

X = USR n (引き数) (nはDEFUSRのnと一致させる)

引き数の型は、整数型、単精度実数型、倍精度実数型、文字型とある。引き数が渡されると、Aレジスターには引き数の型 (FACすなわち浮動小数点アキュムレーターの必要バイト数) が入る。HLペアレジスターには、引き数が格納されているFACへのポインターの番地が入る。引き数が文字の場合には、DEペアレジスターがストリングディスクリプターの番地を指す。

引き数の型		整数型	単精度実数型	倍精度実数型
Aレジの値		2	4	8
浮動小数点アキュムレーターはここを指す	FAC7			仮数の最下位8ビット
	FAC6			仮数の第6位8ビット
	FAC5			仮数の第5位8ビット
	FAC4			仮数の第4位8ビット
	FAC3	引き数の下位8ビット	仮数の下位8ビット	仮数の第3位8ビット
	FAC2	引き数の上位8ビット	仮数の中位8ビット	仮数の第2位8ビット
	FAC1		仮数の上位8ビット	仮数の最上位8ビット
	FAC0		指数部	指数部

HLレジスターは常にFAC3の番地を指す。実数型は浮動小数点形式になっている。また、ストリングディスクリプターは次のようになっている。

引き数の型	文 字 型
Aレジの値	3
DEレジスター → 一はこの 番地をさす	文字の長さ 8ビット
	実際に文字が格納されている番地の下位8ビット
	実際に文字が格納されている番地の上位8ビット

(例) 任意の整数に5を足す (整数型引き数の例)

```
10 INPUT A%
20 B%=A%+5
30 PRINT B%
```

上のプログラムの20行を機械語サブルーチンにしてみる。

```
10 CLEAR 300, &HFFFF
20 DEFUSR4=&HE000
30 FOR J=0 TO 9
40 READ M:POKE &HE000+J, M
50 NEXT J
60 DATA &H7E, &HC6, &H05, &H77, &H23
70 DATA &H7E, &HCE, &H00, &H77, &HC9
80 INPUT A%
90 B%=USR4 (A%)
100 PRINT B%
```

30行～70行は、モニターモード (MON) で*SE000として入力してもよい。初心者のために機械語の説明をつけておく。

7E→LD A, (HL)	HLレジスターの指す番地の内容 (A%の下位8ビットが入っている) をAレジスターへ入れる
C605→ADD A, 5	Aレジスターに5を加える
77→LD (HL), A	Aの値をHLレジスターの指す番地へもどす
23→INC HL	HLレジスターを+1する
7E→LD A, (HL)	+1されたHLレジスターの指す番地の内容 (A%の上位8ビットが入っている) をAレジスターへ入れる
CE00→ADC A, 0	下位ビットの演算結果のキャリー部分を加えてやる
77→LD (HL), A	Aの値をHLレジスターの指す番地へもどす
C9→RET	サブルーチンの終わり

詳細は、機械語→ニーモニック変換表をみよ。

(例) キーボードから入力した文字を画面表示する (文字型引き数の場合)
N-BASICの画面表示ルーチン (&H0257~&H02CA番地) を使う。すなわち、サブルーチンの中からさらにサブルーチンへ飛ぶ。

```
10 CLEAR 300, &HDFFF
20 DEFUSR3=&HE000
30 INPUT A$
40 X$=USR3 (A$)
```

機械語の部分は、モニターモード (MON) にして*SE000から、13 1A 6F 13 1A 67 7E CD 57 02 C9と入力して、すんだらCTRLキーと「B」を押してBASICモードへもどる。

13→INC DE	DEを1番地進める。その番地に文字が格納されている番地の下位8ビットがある
1A→LD A (DE)	Aレジスターへ
6F→LD L, A	Lレジスターへ取り出す
13→INC DE	さらに1番地進める
1A→LD A (DE)	上位8ビットをAレジスターへ
67→LD H, A	Hレジスターへ取り出す
7E→LD A, (HL)	1文字目をAレジスターへ取り出す
CD5702→CALL 0257	画面出力ルーチンへ
C9→RET	サブルーチンの終わり

(例) 画面をプリントする → COPY
PC-8001は、ターミナルモード時の画面プリント機能 (&H124A番地) を持っている。もともとある機能だから、CLEARで場所を確保しなくてもよい。

```
DEFUSR2=&H124A 画面プリントサブルーチンの入口アドレスを定義する。
X=USR2 (0)      画面プリントサブルーチンを実行する。
```

使用上の注意点

- *この機能は、文字しかできない。ドットグラフィックは、空白となる。
- *画面にカーソル移動で表示したものは、そのままでは左に詰まってプリントされてしまう。これを避けるには、COLOR文のヌルキャラクターコードで0 (ヌル) ではなく、32 (スペース) を指定し、画面にスペースを埋めておく。

ポイント

ファンクション・キーに X=USR2 (0) を定義しておくると便利。ダイレクトモードで次のコマンドを実行すれば、画面プリントができる。

```
DEFUSR2=&H124A
KEY1, "X=USR2 (0) " + CHR$ (13)
```

以後 f・1 を押せば、画面プリントされる。

*ユーザールーチン使用上の注意点

① 引数と結果の型が違う場合

引数が整数でない場合でも、BASIC内のMAKINTと呼ばれるルーチンを経由してリターンすることにより、整数値を返すことができる。

返したい整数値はHLレジスターに入れておく。

PUSH HL	HLレジスターの内容を、スタックにいれる
LD HL, (0024)	HLレジスターに、MAKINTルーチンのアドレスをいれる
	(0024)に、MAKINTルーチンのアドレス279Cが入っている
EX (SP), HL	スタックの内容と、HLレジスターの内容とをいれかえる。これで、HLレジスターには返す値が、スタックにはMAKINTルーチンのアドレスが入る
RET	MAKINTルーチンのアドレスをリターンアドレスとみなし、ジャンプする

② 文字列引数として文字定数を渡すとき

文字定数はプログラム内にあるため、これに対する操作はプログラム中の文字定数を変化させることになる。これを避けるためには、次のように式の型にすることにより、いったん文字領域へコピーしてから引数とする。A\$="ABCD"+" "

③ ユーザールーチンのエラー

ユーザールーチンはBASICとは全く別に動くため、エラーメッセージは表示されない。そればかりでなく、BASICで書かれたプログラムをこわしてしまうこともある。

当然、STOPもきかないため、リセットしか止める方法がない。そこで、プログラムの中に、デバッグ用として画面表示を呼び出す命令を入れておき、どこまでプログラムが動いたかを見ながらまとまった仕事単位にデバッグしていくとよいだろう。

④ 機械語サブルーチンでのスタック

BASICは、ユーザールーチン用スタックとして自動的に8レベル、16バイトを確保し、スタックポインターをセットしてくれる。これより大きいスタックが必要な場合には、自分でスタックをつくらなくてはならない。

E000H番地からE0FFH番地までの256バイトを使うなら128レベルのスタックとなる。

機械語サブルーチンの始めでスタックポインターを退避し、スタックポインターにはE100Hをロードする。これは、PUSH命令などのスタックを使う命令がスタックポインターの指すひとつ手前の番地からスタックを使うため。(PUSH命令の説明参照 104ページ)

機械語サブルーチンからBASICに戻るには、機械語のRET命令を実行する。ただし、RET命令で使われる戻り番地はもとのスタックにあるので、スタックポインターに始めに退避した値をロードしなければならない。

*スタックの作り方

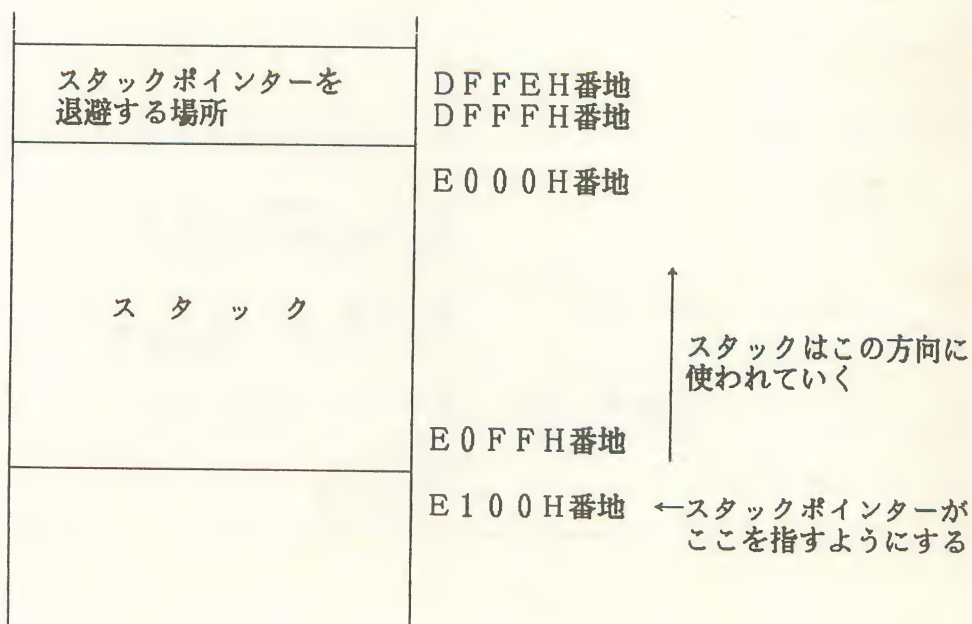
ニーモニックで書くと次のようになる

```
LD    (0DFFE H), SP    スタックポインターを退避する
LD    SP, 0E100 H      スタックポインターを新しいスタックにセ
                        .   ットする
                        .   ・サブルーチン本体
                        .
LD    SP, (0DFFE H)     スタックポインターを復帰する
```

スタックをE000H番地からE0FFH番地につくる。
スタックポインターをDFFE H番地, DFFF番地に退避する。

機械語で書くと次のようになる

```
ED73 FEDF    LD    (0DFFE H), SP
3100 E1      LD    SP, 0E100 H
            .
            .   ・サブルーチン本体
            .
ED7B FEDF    LD    SP, (0DFFE H)
              (機械語では上位下位バイトが逆にな
              るのでFEDF。104ページ参照)
```

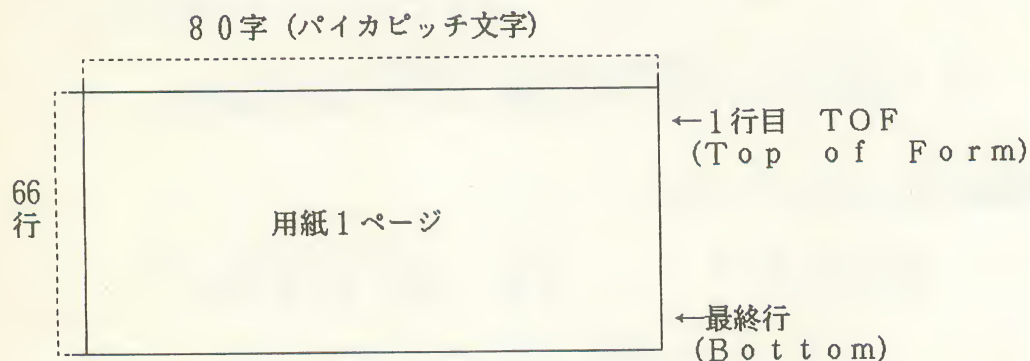


プリンター (PC-8023) の使い方

<基本事項>

- (1) コンピューター本体とプリンターとが正しく接続されていること。
- (2) 電源が入っていること。
- (3) オンライン状態であること (セレクト (SEL) スイッチが押されている)。
- (4) 用紙が正しくセットされていること。

<印字文字数>



<基本操作>

命 令	機 能
LLIST	プログラムのリストをプリンターへ出力する
LPRINT	データや計算結果をプリンターへ出力する
LPOS (x)	x はダミー変数。プリンターヘッドの現在位置 (横方向) を与える
LPRINT USING	使い方は, PRINT USING と同じで, プリンターへ出力する
LPRINT CHR\$ (12)	改ページして次の TOF へ
LFILLES	使い方は, FILES と同じ。プリンターへ出力する

＜文字制御＞ パワーオン時には、パイカで英数記号とカタカナに設定されている。
縮小文字の強調印字はできない。

機 能	命 令
パイカにする	LPRINT CHR\$ (27) ; CHR\$ (78) ;
エリートにする	LPRINT CHR\$ (27) ; CHR\$ (69) ;
縮小文字にする	LPRINT CHR\$ (27) ; CHR\$ (81) ;
プロポーショナル	LPRINT CHR\$ (27) ; CHR\$ (80) ;
英数記号とカタカナ	LPRINT CHR\$ (27) ; " \$" ;
英数記号とひらがな	LPRINT CHR\$ (27) ; " & " ;
拡大文字にする	LPRINT CHR\$ (14) ;
拡大文字の解除	LPRINT CHR\$ (15) ;
強調文字にする	LPRINT CHR\$ (27) ; " !" ;
強調文字の解除	LPRINT CHR\$ (27) ; CHR\$ (34) ;
アンダーラインを引く	LPRINT CHR\$ (27) ; CHR\$ (88) ;
アンダーラインの解除	LPRINT CHR\$ (27) ; CHR\$ (89) ;
ドットをプリントする	LPRINT CHR\$ (27) ; CHR\$ (83) ; " n " nは送るバイト数で4桁の10進数

ドットを使って漢字の「江」をプリントするには、まず16×16のマス目に文字パターンを作る。マス目の1つが1ドット。ぬりつぶされたマスがビットのON、空白がOFFと考え、縦8マス分を8桁の2進数とする。左から右へ16回やれば文字の上半分のDATAができる。SUBROUTINEでデータをPRINTERへ送り、上半分を印字する。改行して下半分を印字する。小漢字にしたければ、縦16マスを奇数マスと偶数マスで別々にDATAを作り、10行の改行幅を04とする。奇数マスが先にうたれ、2ドット改行、偶数マスがうたれる。

```

10 LPRINT CHR$ (27) ; " T " ; " 16 " ;
20 FOR J =1 TO 2 : FOR K =1 TO 16
30 READ A : LPRINT CHR$ (27) ; CHR$ (83) ; " 0001 " ; : GOSUB 500
40 NEXT K : LPRINT : NEXT J : END
50 DATA 0,64,130,4,8,0,0,4,4,4,4,252,4,4,4,4
60 DATA 0,32,64,192,32,16,0,64,64,64,64,127,64,64,64,64
500 OUT &H10,A : OUT &H40,0 : OUT &H40,1
510 S = INP (&H40) AND 1 : IF S=0 THEN RETURN ELSE 510

```

小漢字のDATAは、
50 DATA 0,0,66,0,68,0,34,2,2,2,126,2,2,2,2,0
60 DATA 0,41,8,74,8,32,64,64,64,64,126,64,64,64,64,0

<幅と行と位置の制御>

機 能	命 令																
改行幅を1/6 インチにする	LPRINT CHR\$ (27) ; "A" ;																
改行幅を1/8 インチにする	LPRINT CHR\$ (27) ; "B" ;																
n/144イン チの改行幅にす る	LPRINT CHR\$ (27) ; "T" ; "n" ; n=01~99, また2/144は1ドット幅となる																
水平タブを設定 する	<p>LPRINT CHR\$ (27) ; CHR\$ (40) ; "a, b , c,p. " ; a~pは3桁の10進数。10は010と書くこと コンマ (,) は継続コード, ピリオド (.) は終了コード a~pは16個まで設定できる (例)</p> <pre> 100 LPRINT CHR\$ (27) ; CHR\$ (40) ; "010, 020, 030. " ; 110 FOR I=1 TO 4 120 LPRINT I ; 130 GOSUB 500 140 NEXT I 150 END </pre> <p>実行結果</p> <table border="1" style="margin: 10px auto; border-collapse: collapse;"> <tr> <td style="text-align: center; width: 20%;">1</td> <td style="text-align: center; width: 20%;">2</td> <td style="text-align: center; width: 20%;">3</td> <td style="text-align: center; width: 20%;">4</td> </tr> <tr> <td colspan="4">←10→</td> </tr> <tr> <td colspan="4">←-----20-----→</td> </tr> <tr> <td colspan="4">←-----30-----→</td> </tr> </table> <p>プリントする場合, 以下のサブルーチンを実行し, 水平タブコ ードをプリンターへ送っておくこと</p> <pre> 500 OUT &H10, &H9: OUT&H40, 0: OUT &H40, 1 510 D=INP (&H40) AND 1: IF D=0 THEN 520 ELSE 510 520 RETURN </pre>	1	2	3	4	←10→				←-----20-----→				←-----30-----→			
1	2	3	4														
←10→																	
←-----20-----→																	
←-----30-----→																	

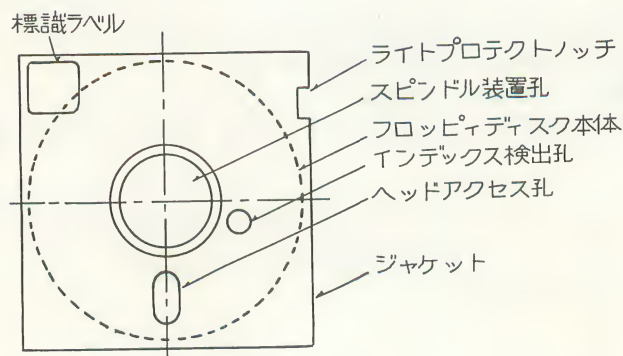
機 能	命 令
水平タブを全面解除する	LPRINT CHR\$ (27) ; CHR\$ (50) ;
水平タブを部分解除する	LPRINT CHR\$ (27) ; CHR\$ (41) ; " a, b . " ; (例) 前ページの例のうち、10と20の水平タブを解除する LPRINT CHR\$ (27) ; CHR\$ (41) ; " 010 , 020. " ;
レフトマージンを設定する	LPRINT CHR\$ (27) ; " L " ; " n " ; nは3桁の10進数で記入する (例) LPRINT CHR\$ (27) ; " L " ; " 010 " ; <div style="border: 1px solid black; padding: 10px; margin: 10px 0;"> <div style="display: flex; align-items: center;"> <div style="border-right: 1px dashed black; padding-right: 10px;">←10→</div> <div style="margin-left: 10px;">←ここからプリントする</div> </div> </div>
レフトマージンを解除する	LPRINT CHR\$ (27) ; " L000 " ; Lの後にゼロを3個書けばよい
n行改行する	LPRINT CHR\$ (31) ; CHR\$ (16+n) ; 1=<n=<15のこと
ドットスペースをプリントする	LPRINT CHR\$ (27) ; CHR\$ (n) ; 1=<n=<6のこと。 プロポーショナル文字のときのみ6ドットまでスペースがとれる

ミニフロッピーディスク

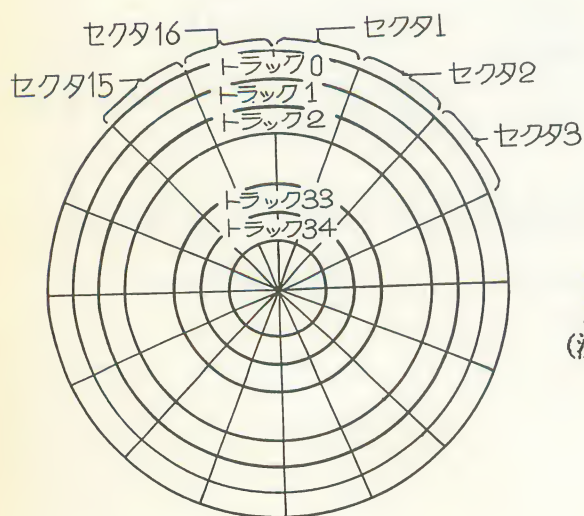
<基本事項>

PC-8001の外部記憶装置としてミニフロッピーディスク(PC-8031, 8032)がある。PC-8032は、PC-8031と接続して使う拡張ディスクユニットで、両方とも2台のミニフロッピーディスクドライブが入っている。PC-8001とPC-8031の接続には、拡張ユニット(PC-8011)、I/Oユニット(PC-8012)、I/Oポート(PC-8033)のいずれかを用いる。

*フロッピーディスクの外形



*ディスクの構造



- 1 セクター = 256 バイト (256 文字)
- 8 セクター = 1 クラスタ
- 2 クラスタ = 1 トラック
- 35 トラック = ディスク 1 枚

16 セクター × 35 = 560 セクター
 560 セクター × 256 = 143360 バイト
 (注) ユーザーが使うことができるのは、
 139264 バイト。

<ディスクファイルの管理>

フロッピーディスクの読み書きは、1セクター単位で行われる。したがって、トラック番号とセクター番号で物理的に読み書きする場所が決められる。通常は、特別に意識しなくても、ファイル名さえ与えてやれば、自動的に読み書きする。管理は、1クラスター単位でBASICが行っている。

項目の集まりをレコードといい、レコードの集まりをファイルという。

<トラックの使用割り当て>

*システムディスク (DISK BASICが記憶されているディスク)

トラック	セクター	内 容
0	1	IPL
0	2~16	DISK BASIC
1~2	すべて	
3~17	すべて	ユーザーが使える領域
18	1~12	ディレクトリー
18	13	ID
18	14~16	FAT
19~34	すべて	ユーザーが使える領域

*データディスク (DISK BASICを含まないディスク)

トラック	セクター	内 容
0~17	すべて	ユーザーが使える領域
18	1~12	ディレクトリー
18	13	ID
18	14~16	FAT
19~34	すべて	ユーザーが使える領域

<ディレクトリー>

ディレクトリーとは、ディスクに記憶されているファイルの名前、属性、どの場所から記憶されているかを示す登録簿のこと。ファイルごとに1つ作られ、1つのディレクトリーは16バイトとられる。

ディレクトリーの構造

バイト	内 容
0～5	ファイル名 先頭バイトが16進数のFFならば、未使用のディレクトリー 0ならば、削除されたファイル
6～8	拡張子
9	属性を示す 0：アスキー形式 10：アスキー形式、書き込み禁止 40：アスキー形式、リード・アフター・ライト 80：非アスキー形式 90：非アスキー形式、書き込み禁止 C0：非アスキー形式、リード・アフター・ライト
10	ファイルの先頭クラスター番号
11～15	未使用

<FAT>

FAT（ファイル・アロケーション・テーブル）は、各クラスターの使用・未使用状態を示すもの。ファイルは普通、物理的につながったクラスター上に作られるわけではなく、離れたクラスターをつないで構成される。MOUNTコマンドを実行すると、FATがPC本体のメモリーにコピーされる。FATは、14～16の各セクターにとられ、それぞれのセクターの内容はすべて同一となっている。各セクターの0～69バイトがそれぞれのクラスターに対応し、1バイトでクラスターの使用状態を表す。

FATの値	ク ラ ス タ ー の 状 態
0～45	使用中を示す 後続クラスターの番号が入っている
C1～C8	使用中を示す 連続するクラスターの最後 下位4ビットには、そのクラスターで実際に使われているセクターの数が入っている
FE	使用できない DISKBASICやディレクトリー等で使っている
FF	未使用

<DISK BASICのスタート>

- (1) ディスクユニットの電源を入れる。
- (2) システムディスクをドライブ1に入れて、ふたをしめる。
- (3) PC-8001の電源を入れる。
- (4) システムプログラムが読み込まれる。
- (5) 画面に次の表示が出る。

DISK Version (14-NOV-1980)
How many files (0-15) ?

- (6) 同時にオープンできるファイルの数は15までなので、15以下の数字を入力すると、OKと出て、準備完了。

<バックアップファイルの取り方>

ディスクに記憶されているファイルは、使用中のミスによりこわれてしまうことがある。重要なファイルが記憶されているディスクは、あらかじめ複写しておいたほうがよい。システムディスクは、ディスクユニットを買うと1枚しかついてこないで、特に必要。

*システムディスクのバックアップ作成

- (1) ドライブ1にシステムディスクを装着し、ドライブ2にバックアップディスクを装着する。
- (2)

```
mount1          (RET)
OK
run"backup"     (RET)
Back up a disk
Mount master disk on drive1, then
hit return
(RET)
Mount new disk on drive2, then
hit return
(RET)
Format disk (y/n) ?
ディスクのフォーマットが必要ならば、y (RET)。
必要なければ、n (RET)。
```
- (3) 最後にCompleteとでて、完了。

(注) (RET) は、RETURNキーを押すことを示す。

データディスクのバックアップファイルも同様にして作ることができる。

- (1) でシステムディスクのかわりにデータディスクを装着すればよい。

<フォーマットの仕方>

ミニフロッピーディスクをPC-8031で使い、入出力ができるようにする作業をディスクのフォーマッティングという。

レベル	実行方法	内 容
1	FORMATコマンドを使う	BASICのDSKI\$, DSKO\$でのみ、読み書きできるようになる。FAT, ディレクトリー, IDの初期化はされないの、通常のファイルを作ることはいできない
2	システムディスクのformatプログラムを使う	FAT, ディレクトリー, IDの初期化をする BASICのコマンド, ステートメントを使って、ファイルを作ることができるようになる

*formatプログラムの実行

- (1) システムディスクからformatプログラムをロードし、実行する。
- (2) create system disk (y/n) ?

システムディスクを作るなら y (RET), データディスクを作るなら n (RET) と答える。

- (3) mount new disk on drive 2
sure (y/n) ?

- (4) フォーマッティングするディスクをドライブ2へ装着し、y (RET)。
フォーマッティングされる。データディスクを作るのであればこれで終了。
システムディスクを作るときは、さらに以下の表示が出る。
mount system disk on drive 1
sure (y/n) ?

システムディスクをドライブ1に装着し、y (RET)。

- (5) Completeと出て、すべて終了。

<DISK BASICのバージョンの違い>

- ① DISK BASICの新しいバージョンは、スタート時に

Disk Version [14—Nov—1980]

と表示される。古いバージョンは [] 内の表示はない。

- ② 古いバージョンでは、メモリー上のFATの内容が変更されたか否かにかかわらず、remove時にFATをディスクに書き込む。
新しいバージョンでは、FATの内容に変更があったときのみremove時にFATをディスクに書き込む。

- ③ ②の理由から、ライトプロテクトノッチに書き込み禁止のラベルをはったディスクに対してremoveを行ったとき、以下ようになる。
- a 古いバージョンでは、FATを書き込もうとして書き込めないでループ状態となる。つまり、removeはできない。
 - b 新しいバージョンでは、FATを書き込まないので、正常に終了する。

データファイル

項目（アイテム）の集まりをレコードといい、レコードの集まりをファイルという。

社員コード	社員名	住所	社員コード	社員名	住所	社員コード	社員名	
項目								
レコード								
ファイル								

データファイルには、次の二種類がある。

① シーケンシャルファイル

「順編成ファイル」とも呼ばれる。レコードを順次にならべていったファイル。10番目にあるレコードを読みたいときでも、1番目のレコードから順番に読んでいかなければならない。また、ファイルの中にあるレコードの内容を削除、変更することはできない。削除、変更するときは、新しいファイルを作る必要がある。レコードの長さは一定ではない。カセットテープのファイルなどは、典型的なシーケンシャル・ファイル。

② ランダムファイル

「乱編成ファイル」とも呼ばれる。レコード番号を指定すれば、読み書きはどのレコードからでもできる。ディスクでのみ、使うことができる。レコードの長さは、256バイトで一定。

<シーケンシャルファイル>

*シーケンシャルファイルに必要な命令

命 令	機 能
MOUNT OPEN	ファイルを開く モード INPUT.....読む APPEND.....追加 OUTPUT.....新規作成
PRINT#	レコードを書き出す
PRINT# USING	書式を指定してレコードを書き出す
INPUT#	レコードを読む
LINE INPUT#	1行をすべて文字列として読む
INPUT\$ 関数	指定した文字をファイルから読む
EOF 関数	ファイルの終わりを検出する
CLOSE	ファイルを閉じる
REMOVE	

*入出力命令

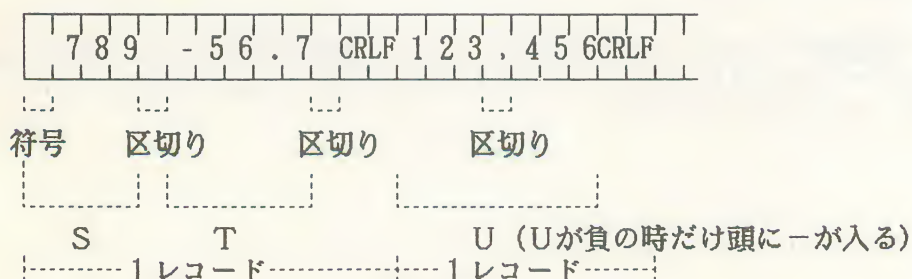
① 数字の入出力 <出力>

```

500 S=789
510 T=-56.7
520 U=123456
530 PRINT#1, S;T
540 PRINT#1, USING "#####", U

```

ディスクの記録のされ方



Uは書式指定して書き出したので、数字の間に“, ”がはいる。”, ”は、項目の区切りとみなされるので、2つの項目に分割される。

CR (キャリッジ・リターン) とLF (ライン・フィード) は、レコードの境界を表す文字で、それぞれ16進数で0Dと0A。

<入力>

前記のファイルを読む場合

```
100 INPUT#1, S, T
110 INPUT#1, U1, U2
120 PRINT S;T;U1;U2
```

結果

789 -56.7 123 456

② 文字の入出力

```
200 S$="マイコン"
210 T$="PC8001"
220 PRINT#1, S$;T$
230 PRINT#1, S$;"", " ";T$
240 PRINT#1, CHR$(34); "マイコン", PC8001
    ";CHR$(34)
```

*ディスクの記録のされ方

マ	イ	コ	ン	P	C	8	0	0	1	C	R	L	F	マ	イ	コ	ン	,
P	C	8	0	0	1	C	R	L	F	"	マ	イ	コ	ン				, P
C	8	0	0	1	"	C	R	L	F									

文字データの場合、項目の区切りとなるのは空白ではなくコンマ (,) である。
文字データの中にコンマまたは、意味のある空白を入れたい場合は、CHR\$(34) (") で囲む。

前記のファイルを以下のようにして読む。最初のレコードは、S\$とT\$がつながって1つの項目となる。(ST\$として読む)

```
100 INPUT#1, ST$
110 INPUT#1, S$, T$: INPUT#1, U$
120 PRINT ST$: PRINT S$, T$
130 PRINT U$
```

結果

マイコンPC8001
マイコン PC8001
マイコン , PC8001

<シーケンシャルファイルのプログラム例1>

*機能

シーケンシャルファイル"セイセキ"を作成する。入力した番号, 名前, 点数で1レコード。入力終了時は, 番号を9999とする。

*ポイント

文字項目を書き出すときは, 間にコンマ(,)を入れる。

```
10  ' *****
20  ' *
30  ' *   シーケンシャル   レイ   (1)   *
40  ' *
50  ' *****
60  OPEN  "セイセキ"  FOR OUTPUT AS #1
70  INPUT "ハ`ンコ`ウ" ; B$
80  IF B$="9999" GOTO 130
90  INPUT "ナマエ" ; N$
100 INPUT "テンスウ" ; T
110 PRINT#1, B$ ; ", " ; N$ ; ", " ; T
120 GOTO 70
130 CLOSE#1
140 END
```

<シーケンシャルファイルのプログラム例2>

*機能

例(1)で作成したファイルを読み, 画面上に番号, 名前, 点数を表示する。

*ポイント

EOF(1)でファイルの終了を判断する。

```
10  ' *****
20  ' *
30  ' *   シーケンシャル   レイ   (2)   *
40  ' *
50  ' *****
60  OPEN  "セイセキ"  FOR INPUT AS#1
70  IF EOF(1) GOTO 110
80  INPUT#1, B$, N$, T
90  PRINT B$, N$, T
100 GOTO 70
110 CLOSE#1
120 END
```

<シーケンシャルファイルのプログラム例3>

*機能

例(1)で作成したファイルにデータを追加する。追加データが終わりのときは、番号を9999とする。

*ポイント

シーケンシャルファイルにレコードを追加するときのモードは、APPEND。

```

10  ' *****
20  ' *
30  ' *   シーケンシャル   レイ   (3)   *
40  ' *
50  ' *****
60  OPEN "セイセキ" FOR APPEND AS #1
70  INPUT "ハンコウ" ; B$
80  IF B$="9999" GOTO 130
90  INPUT "ナマエ" ; N$
100 INPUT "テンスウ" ; T
110 PRINT#1, B$ ; ", " ; N$ ; ", " ; T
120 GOTO 70
130 CLOSE#1
140 END

```

<ランダムファイル>

*ランダムファイルに必要な命令

命 令	機 能
MOUNT	
OPEN	ファイルを開く
GET	ディスクからバッファへ読み込む
PUT	バッファからディスクへ書き出す
FIELD	バッファの項目を定義する
LSET	バッファへデータを入れる
RSET	
MKI\$関数	
MKS\$関数	数字を文字へ変換する
MKD\$関数	
CVI関数	
CVS関数	文字を数字へ変換する
CVD関数	
LOF関数	最大のレコード番号を与える
LOC関数	PUTまたはGETしたレコード番号を与える
CLOSE	ファイルを閉じる
REMOVE	

<バッファについて>

ランダムファイルの場合、ディスクとの入出力にバッファという作業領域を必要とする。バッファは、ランダムファイルの1レコードの長さに等しい256バイトで、15個まで使うことができる。DISK BASICを開始する際、How many files (0-15)? という問いには、同時にオープンするファイルの数を答える。この入力によって使うことのできるバッファの数が決まる。

OPEN, GET等で指定するファイル番号によって、使用するバッファの番号が決まる。バッファ内の項目の定義は、FIELD文でおこなう。ランダムファイルの場合、入出力はすべて文字項目として扱う。数字項目を書き出すときは、MKS\$, MKI\$, MKD\$で文字項目に変換してバッファへ入れる。数字項目を読み込むときは、CVS, CVI, CVDで数字項目に変換して使う。

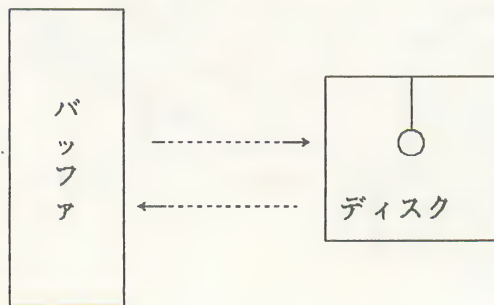
(数値変数)

MKI\$, MKS\$, MKD\$
LSET, RSET

CVI, CVS, CVD

(文字変数)

LSET, RSET



<ランダムファイルプログラム例(1)>

*機能

ランダムファイル"ウリアゲ"を作成する。入力項目は、社員コード、名前、商品A・B・Cそれぞれの売り上げ金額。入力終了のときは、社員コードを99999と入れる。

*ポイント

数字は、MKS\$ (MKシリーズ関数) で文字項目に変換し、バッファへ送る。

```

10 ' ***   ランダム   レイ   (1)   ***
20 OPEN   "ウリアゲ"   AS #1
30 FIELD  #1, 6   AS  X$, 14   AS  Y$, 4   AS  A$,
      4   AS  B$, 4   AS  C$
40 INPUT  "シャインコート" ; P$
50 IF P$ = "999999" GOTO 130
60 INPUT  "ナマエ" ; Q$
70 INPUT  "A ノ ウリアゲ" ; A
80 INPUT  "B ノ ウリアゲ" ; B
90 INPUT  "C ノ ウリアゲ" ; C
100 LSET  X$ = P$ : LSET  Y$ = Q$
110 LSET  A$ = MKS$ (A) : LSET  B$ = MKS$ (B) :
      LSET  C$ = MKS$ (C)
120 PUT #1 : GOTO 40
130 CLOSE #1 : END

```

<ランダムファイルプログラム例 (2) >

*機能

例 (1) で作成したファイルを読み、指定した社員コードの売り上げを画面上に出
力する。処理終了のときは、社員コードを999999と入力する。

*ポイント

レコードナンバーと社員コードの対応をとるため、あらかじめすべてのレコードを
読んで対応表をつくっている。

```

10  ' *****
20  ' * ランダム レイ (2) *
30  ' *****
40  OPEN "ウリアケ" AS #1
50  FIELD #1, 6 AS X$, 14 AS Y$, 4 AS A$,
      4 AS B$, 4 AS C$
60  DIM N (10), M$ (10)
70  FOR I=1 TO LOF (1)
80      GET #1
90      N (I) = LOC (1) : M$ (I) = X$
100  NEXT I
110  INPUT "シャイン コート "; S$
120  IF S$ = "999999" GOTO 220
130  FOR I=1 TO LOF (1)
140      IF M$ (I) = S$ GOTO 170
150  NEXT I
160  PRINT "カイトウ シャインコート ナシ!"; GOTO 110
170  GET #1, N (I)
180  PRINT S$;
190  PRINT USING "&" & " "; Y$;
200  PRINT USING "#####" "#####";
      ; CVS (A$); CVS (B$); CVS (C$)
210  GOTO 110
220  CLOSE #1
230  END

```

*特殊な I/O

いままで説明してきた命令によるファイルの入出力は、FATによって管理される。
DSKI\$, DSKO\$は、FATとは関係無しにドライブ番号、トラック番号、セク
ター番号を指定することにより、ディスク上の任意の場所に対して入出力することがで
きる。マウント命令も必要ない。

任意の場所の内容を読む.....DSKI\$
任意の場所へ書き出す.....DSKO\$

DSKI\$, DSKO\$の入出力は、0番のバッファを使用する。
使い方を誤るとディスク上のファイルを壊してしまうことがあるので、使うときは、
十分な注意が必要。

<DSKI\$のプログラム例>

*機能

ディスクの使用状況をプリンターに出力する。ディスクを装着して、ドライブ番号に答えるだけでよい。MOUNT命令は必要ない。

*ポイント

行番号100~120までで、FIELD文を配列定義している。

DSKI\$でディレクトリー、FATを読んでいる。バッファの番号は0。

```

10  ' *****
20  ' *   ディスク   カンリヒョウ   サクセイ   *
30  ' *****
40  CLEAR1000
50  DIM N$ (15), Z$ (15), C$ (15)
60  LPRINT " ディスク   カンリヒョウ "; LPRINT
70  INPUT " トライブ   ナンバー   ハ "; D
80  LPRINT " トライブ   ナンバー   ハ? "; D
90  FAT$=DSKI$ (D, 18, 14)
100 FOR I=0 TO 15
110     FIELD#0, I*16 AS DM$, 9 AS N$ (I),
        1 AS Z$ (I), 1 AS C$ (I)
120 NEXT I
130 LPRINT " ファイルメイ           ソクセイ   ショウクラスター "
140 FOR I=1 TO 12
150     DM$=DSKI$ (D, 18, I)
160     FOR J=0 TO 15
170         IF ASC (N$ (J)) = 0 OR
            ASC (N$ (J)) = &HFF THEN 270
180         Z=ASC (Z$ (J)) : C=ASC (C$ (J))
190         F$=" "
200         IF (Z AND &H80) = 0 THEN F$=F$+"A"
210         IF (Z AND &H10) > 0 THEN F$=F$+"P"
220         IF (Z AND &H40) > 0 THEN F$=F$+"R"
230         LPRINT USING "&          & &&          ##"
            ; N$ (J), F$, C;
240         C=ASC (MID$ (FAT$, C+1))
250         IF C<=69 THEN LPRINT USING "->##"
            ; C; : GOTO 240
260     LPRINT
270 NEXT J, I
280 FOR C=0 TO 69
290     S=ASC (MID$ (FAT$, C+1))
300     IF S=&HFF THEN FR=FR+1
310 NEXT C
320 LPRINT " ノコリ   ハ "; FR; "   クラスター ";
        FR*8*256; "   ハイト "
330 END

```


エラーメッセージ

エラー

コード エラーメッセージ

- 1 N E X T w i t h o u t F O R
FORがない。NEXTとFORが対応していない。
- 2 S y n t a x e r r o r
文法違反。LIST. で調べよ。
- 3 R E T U R N w i t h o u t G O S U B
GOSUBとRETURNが対応していない。RET
URNだけある。
- 4 O u t o f D A T A
READ文で読むべきデータがDATA文にない。
- 5 I l l e g a l f u n c t i o n c a l l
ステートメントの使い方がまちがっている。あるいは
限度を越えている。
- 6 O v e r f l o w
数値や演算結果が許容範囲を越えた。
- 7 O u t o f m e m o r y
プログラムや配列が大きすぎてメモリー不足。
- 8 U n d e f i n e d l i n e n u m b e r
プログラム行が定義されていない。
- 9 S u b s c r i p t o u t o f r a n g e
配列変数の添字が規定範囲にない。
- 10 R e d i m e n s i o n e d a r r a y
同じ名前の配列を再定義した。
- 11 D i v i s i o n b y z e r o
0で割った。あるいは、未定義の変数で割った。
- 12 I l l e g a l d i r e c t
ダイレクトモードで使えないコマンドが入った。
- 13 T y p e m i s m a t c h
式の左右の変数の型が合っていない。
- 14 O u t o f s t r i n g s p a c e
ストリング領域不足。CLEAR文を調べよ。

- 15 `String too long`
引用符 (" ") で囲まれた文字が255文字を越えた。
- 16 `String formula too complex`
文字式が複雑すぎる。
- 17 `Can't continue`
エラー停止かブレーク後、プログラムを変更したか、あるいはプログラムが存在しないためCONTできない。
- 18 `Undefined user function`
DEF文で定義する前にユーザー関数を呼んだ。
- 19 `No RESUME`
エラー処理に入ったが、RESUMEがない。
- 20 `RESUME without error`
エラーがないのにRESUMEがある。
- 21 `Unprintable error`
エラーメッセージに定義されていないエラー。通常、未定義のエラーコードを含むERROR文で起こる。
- 22 `Missing operand`
ステートメントの中に必要となるパラメーターなどがない。
- 23 `Line buffer overflow`
行を入力するとき、長さの有効範囲を超えた。通常は超えた部分は無視され、エラーメッセージは出ない。
- 24 `Position not on screen`
指定したカーソル位置などが、画面の範囲外。
- 25 `Bad File Data`
ファイルにあるデータの形式がまちがっている。
- 26 `Disk BASIC Feature`
ディスクがつながっていないのに、Disk BASICの命令を実行。
- 27 `Communications Buffer Overflow`
周辺機器との入出力のためのバッファがオーバーフローした。
- 28 `Port not initialized`
インタフェース用LSIの初期設定がされていない。

29 Tape read ERROR

カセットテープがうまく読み取れない。

<DISK BASICのエラー>

50 FIELD overflow

FIELD文で257バイト以上の文字を割り当てた。

51 Internal error

BASIC内部でのエラー。

52 Bad file number

オープンされていないファイル・ナンバーをアクセスした。

53 File not found

LOAD, KILL, OPENなどで存在しないファイルをアクセスした。

54 Bad file mode

シーケンシャル・ファイルでOPENしたファイルにランダム・アクセスした。

55 File already open

すでに開かれているファイルにOPENやKILLした。

56 Disk not mounted

mountしていないディスクをアクセスした。

57 Disk I/O error

ディスクにREAD・WRITE errorが生じ、読み直しても修正できなくなった。

58 File already exists

NAME文によって定義されたファイル名がすでに登録されている。

59 Disk already mounted

mountされているディスクに対してまたmountした。

60 Disk full

ディスク上のすべての場所を使いきった。

61 Input past end

ファイルのすべてのデータを読みつくした後にinput文が実行された。

- 62 B a d f i l e n a m e
 不適當なファイルネームを使った。
- 63 D i r e c t s t a t e m e n t i n f i l e
 アスキー・フォーマットでのLOAD中にダイレクト
 ・ステートメントがあった。
- 64 B a d a l l o c a t i o n t a b l e
 FATがこわれた。
- 65 B a d d r i v e n u m b e r
 不適當なドライブネームを指定した。
- 68 R e n a m e a c r o s s d i s k s
 ファイル名のつけかえで、違ったドライブナンバーを
 指定した。
- 71 F i l e n o t O P E N
 まだ開かれていないファイルに対してアクセスした。
- 72 F i l e w r i t e p r o t e c t e d
 ライト・プロテクトがかかっているファイルに対して
 書き込んだ。

PC-8000シリーズ ラインアップ

型 名	品 名	特 長
PC-8001	パーソナル コンピューター	8ビット, パーソナルコンピューター
PC-8006	増設RAM	8001, 8011の増設用IC セット
PC-8011	拡張ユニット	PROM, RAM, RS-232C インタフェース, パラレルポ ート, リアルタイムクロック, ミ ニフロッピーディスクインタフ ェースを備えた拡張ユニット
PC-8012	I/Oユニット	リアルタイムクロック, ミニフ ロッピーディスクインタフ ェース, 各種拡張用ボード差し込み用ソ ケットを備えたユニット
PC-8012-01	8012用ユニバーサ ルボード	自作用プリントボード
PC-8023	80桁ドットマトリッ クスプリンター	多様な機能を持った, CPU内蔵 のプリンター ドットグラフィッ ク, ひらがなが使用できる
PC-8031	ミニディスクユニット	CPU内蔵のインテリジェントミ ニフロッピーユニット。143K バイト×2ドライブ実装
PC-8032	ミニディスク拡張ユニ ット	PC-8031の拡張用ドライブ ユニット 143Kバイト×2ド ライブ実装
PC-8033	PC-8031FDC I/Oポート	PC-8031とPC-8001 を, PC-8011, PC-80 12なしで接続するためのI/O ユニット
PC-8031-2W	両面倍密デュアルミニ ディスクユニット	両面倍密度タイプ 320Kバイト×2ドライブ実装
PC-8032-2W	拡張用両面倍密デュ アルミニディスクユニ ット	PC-8031-2Wの拡張用ド ライブユニット 320Kバイト×2ドライブ実装

PC-8000シリーズ ラインアップ

型 名	品 名	特 長
PC-8031-1V	シングルドライブ片面ミニディスクユニット	PC-8031の1ドライブタイプ。143Kバイト×1ドライブ
PC-8031-FD1	拡張用シングルドライブユニット	PC-8031-1Vの拡張用ドライブユニット
PC-8041	12インチ・グリーンディスプレイ	
PC-8042	12インチ・カラーディスプレイ	低解像度
PC-8044	家庭テレビ用カラーアダプター	家庭用カラーテレビをPC-8001に接続するときに使用
PC-8045	ライトペン	
PC-8046	9インチ・グリーンディスプレイ	
PC-8049	12インチ・カラーディスプレイ	高解像度
PC-8062	RS-232Cケーブル	PC-8001をモデム・音響カプラーに接続するときに使用
PC-8091	カラーディスプレイ用ケーブル	PC-8001とPC-8049を接続するときに使用
PC-8092	グリーンディスプレイ用ケーブル	PC-8001に付属
PC-8093	CMT用ケーブル	PC-8001に付属
PC-8094	プリンター用ケーブル	PC-8001とプリンターを接続するときに使用
PC-8095	PC-8011用RS-232Cケーブル	PC-8011をモデム・音響カプラーに接続するときに使用
PC-8096	PC-8011用IEEE-488ケーブル	PC-8011をIEEE-488規格の装置に接続するときに使用

PC-8000シリーズ ラインアップ

型 名	品 名	特 長
PC-8097	GP-IB (IEEE-488) インタフェースセット	コントロール用ファームウェア+ドライブ用バッファ+PC-8096のセット
RM-210	カセットテープレコーダー	プログラムのセーブとロード用

PC-8000シリーズには、拡張用のユニットとして、PC-8011とPC-8012の2種類が用意されている。PC-8011は各種のインターフェースが内蔵されたもので、ケーブルさえつなげばすぐに各装置に接続することができる。

PC-8012は、最小限の機能におさえ、必要に応じてボードを差し込んで使うよう、ソケットが7枚分内蔵されている。

<PC-8011の機能>

1	32KバイトRAM	CP/M, UCSD PASCALなどを使用するとき、N-BASICのROMを切り離し、こちらのRAMを使用する
2	8KバイトPROM	ユーザーのプログラムをROM化して差し込むことができる
3	RS-232Cインタフェース	EIA (Electronic Industrial Association) の決めたシリアルデータの伝送規格に対応したインタフェースで、モデム・音響カプラーが接続できる
4	FDC I/Oポート	ミニディスクユニットを接続するためのI/Oポート
5	汎用 (はんよう) パラレルI/Oポート	8ビット入力ポート、8ビット出力ポート、4ビット入力ポート、4ビット出力ポートがあり、各種入出力機器を接続できる
6	IEEE-488インタフェース	IEEE-488インタフェースバスは、計測器とコンピュータを接続するインタフェースである
7	拡張I/Oバス	ユーザーに開放されたI/O命令実行時の入出力機能と割り込み処理に必要な諸データの入出力機能
8	割り込み制御回路	16レベルプライオリティ付の割り込みコントロール機能
9	リアルタイム割り込み	1.66msの周期でCPUに割り込みをかける機能

<PC-8012の機能>

1	マザーバス	ユーザーが開発したインタフェースカードを挿入するためのソケットが7つある。増設RAMボードが4枚(128Kバイト)まで挿入できる。
2	PROM 2Kバイト	ユーザーのプログラムをROM化して差し込むことができる。
3	FDC I/Oポート	ミニディスクユニットを接続するためのI/Oポート
4	割り込み制御回路	8レベル・プライオリティー付の割り込みコントロール機能。
5	リアルタイム割り込み	1.66msの周期でCPUに割り込みをかける機能

<PC-8012-02 増設RAMボード>

32KバイトのRAMを持ち、PC-8001のマスクROM部分を禁止し、こちらのRAMをアクセスできる。このボードは、バンク切り替えにより4枚まで使用できる。ボードごとにリード制御とライト制御ができ、2つ以上のバンクに同時にデータを書き込むことができる。

<PC-8012-03 音声認識ボード>

あらかじめ登録された話者の話す60語を、認識率98%で音声入力できるもの。プログラム、データの入力の他、ゲーム、ビジネスプログラムの実行を音声で指示できる。

(例) PC-8011へのメモリー増設(BASICエリア拡張)

```

10 DEFUSR0=&HFF80
20 FOR J=0 TO 15
30 READ A$:POKE&HFF80+J,VAL("&H"+A$)
40 NEXT J
50 A=USR(0)
60 DATA 01,00,60,11,00,00,21,00
70 DATA 00,ED,B0,D3,E2,C3,E9,17
80 END

```

PC-8011へRAM32Kバイトを増設した場合、PC本体のRAM32Kバイトと合わせて64Kバイトとなるが、8011側へBASIC24Kバイト分をとっているため、実質的にはユーザーエリアは40Kバイトにしかない。

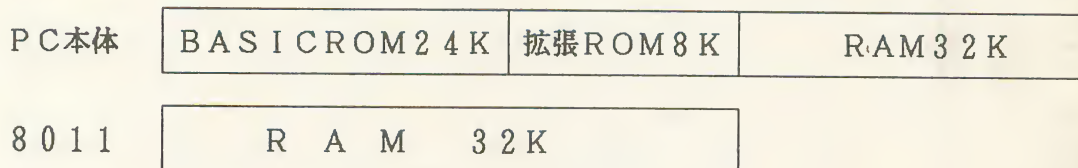
(例) PC-8012へのメモリー増設 (BASICエリア拡張)

```

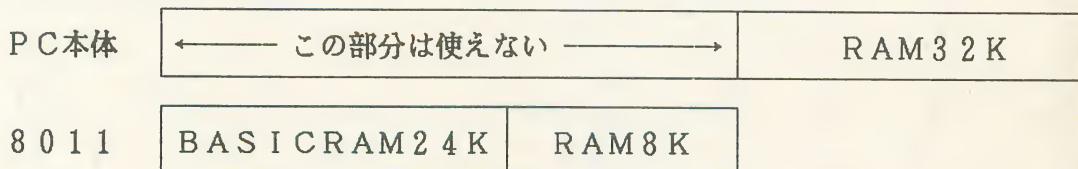
10 DEFUSR0=&HFF80
20 FOR J=0 TO 25
30 READ A$:POKE&HFF80+J,VAL("&H"+A$)
40 NEXT J
50 A=USR(0)
60 DATA 3E,01,D3,E7,3E,10,D3,E2,01,00
70 DATA 60,11,00,00,21,00,00,ED,B0,3E
80 DATA 11,D3,E2,C3,E9,17
90 END

```

これも8011の場合と同じで実質的には8Kバイトしか増設されない。



これが次のようになる。



.....ユーザーが使える領域.....

↑

もし、この24Kバイトもユーザーが使うとすれば、N-BASICは動かないので、すべて機械語でプログラムを書かねばならない。BANKを切り替えながら使うことになる。

<BANK切り替えの仕方>

PCのアドレスの下位32Kバイトは、I/O命令でそっくり切り替えることができる。これをバンク切り替えという。

PCでは下位32KバイトにN-BASICのROMがあるので、バンク切り替えは、機械語でやらなければならない。

PC-8011の場合E2H番地に対するOUT命令を実行する。 出力するデータの値は問わない。
PC-8012の場合バンク0のみ使用した場合、E2H番地に次のデータを出力する。

書き込みのみ許可10H
読み出しのみ許可01H
読み書きとも許可11H
読み書きとも禁止00H

カッコ内のすう
もじ

16進 0 1 2 3 4 5 6 7

キ
ャ
ラ
ク
タ
ー
・
コ
ー
ド
表

1616
進進
数数
には
対横
応が
す上
る位
104
進ビ
ット
た縦
とが
え下
ば位
4B4
はビ
ット
だが
10マ
進ス
では
中75
のに
数字
なる
は

0	0	DE 16	32	0 48	@ 64	P 80	96	p 112
1	SH 1	D 1 17	!	1 49	A 65	Q 81	a 97	q 113
2	SX 2	D 2 18	"	2 50	B 66	R 82	b 98	r 114
3	EX 3	D 3 19	#	3 51	C 67	S 83	c 99	s 115
4	ET 4	D 4 20	\$	4 52	D 68	T 84	d 100	t 116
5	EQ 5	NK 21	%	5 53	E 69	U 85	e 101	u 117
6	AK 6	SN 22	&	6 54	F 70	V 86	f 102	v 118
7	BL 7	EB 23	'	7 55	G 71	W 87	g 103	w 119
8	BS 8	CN 24	(8 56	H 72	X 88	h 104	x 120
9	HT 9	EM 25)	9 57	I 73	Y 89	i 105	y 121
A	LF 10	SB 26	*	: 58	J 74	Z 90	j 106	z 122
B	HM 11	EC 27	+	; 59	K 75	(91	k 107	{ 123
C	CL 12	→ 28	,	< 60	L 76	¥ 92	l 108	! 124
D	CR 13	← 29	—	= 61	M 77) 93	m 109	} 125
E	SO 14	↑ 30	.	> 62	N 78	^ 94	n 110	~ 126
F	SI 15	↓ 31	/	? 63	O 79	— 95	o 111	 127

特殊文字

(読み出したら、戸所定の78〜ことを実行)

例) CHR\$(11) → HOME , CHR\$(13) → RETURN

— 128	⊥ 144	160	— 176	タ 192	ミ 208	= 224	× 240	0
— 129	⌞ 145	。 161	ア 177	チ 193	ム 209	ㄣ 225	円 241	1
— 130	⌏ 146	「 162	イ 178	ツ 194	メ 210	≠ 226	年 242	2
— 131	⌐ 147	」 163	ウ 179	テ 195	モ 211	≡ 227	月 243	3
■ 132	— 148	、 164	エ 180	ト 196	ヤ 212	▲ 228	日 244	4
■ 133	— 149	・ 165	オ 181	ナ 197	ユ 213	▲ 229	時 245	5
■ 134	150	ヲ 166	カ 182	ニ 198	ヨ 214	▼ 230	分 246	6
■ 135	151	ァ 167	キ 183	ヌ 199	ラ 215	▼ 231	秒 247	7
136	「 152	イ 168	ク 184	ネ 200	リ 216	♠ 232	248	8
137	⌑ 153	ウ 169	ケ 185	ノ 201	ル 217	♥ 233	249	9
138	⌒ 154	エ 170	コ 186	ハ 202	レ 218	♦ 234	250	A
■ 139	⌓ 155	オ 171	サ 187	ヒ 203	ロ 219	♣ 235	251	B
■ 140	⌔ 156	ヤ 172	シ 188	フ 204	ワ 220	● 236	252	C
■ 141	⌕ 157	ユ 173	ス 189	ヘ 205	ン 221	○ 237	253	D
■ 142	⌖ 158	ヨ 174	セ 190	ホ 206	・ 222	/ 238	254	E
⊕ 143	⌗ 159	ツ 175	ソ 191	マ 207	。 223	＼ 239	255	F

PRINT CHR\$(n) の使い方

通常は、キャラクターコード表の文字（図形）が表示される。
例外表

n (16進)	コード表の文字	機能
07	BL	BEEPと同じ
09	HT	水平タブ
0A	LF	ラインフィード
0B	HM	カーソルをホームポジションに戻す
0C	CL	画面クリア
0D	CR	キャリッジリターン
1C	→	カーソル移動右
1D	←	カーソル移動左
1E	↑	カーソル移動上
1F	↓	カーソル移動下

キーボードからINKEY\$で入力できる制御コード表

16進コード	キ ー
00	CTRL+@
01	CTRL+A
02	CTRL+B
03	CTRL+C
04	CTRL+D
05	CTRL+E
06	CTRL+F
07	CTRL+G

(例)

```

10 X$=INKEY$
20 IF X$=CHR$(&H01)
   THEN GOTO 100
   .
   .
   .
100 REM***PRG1***

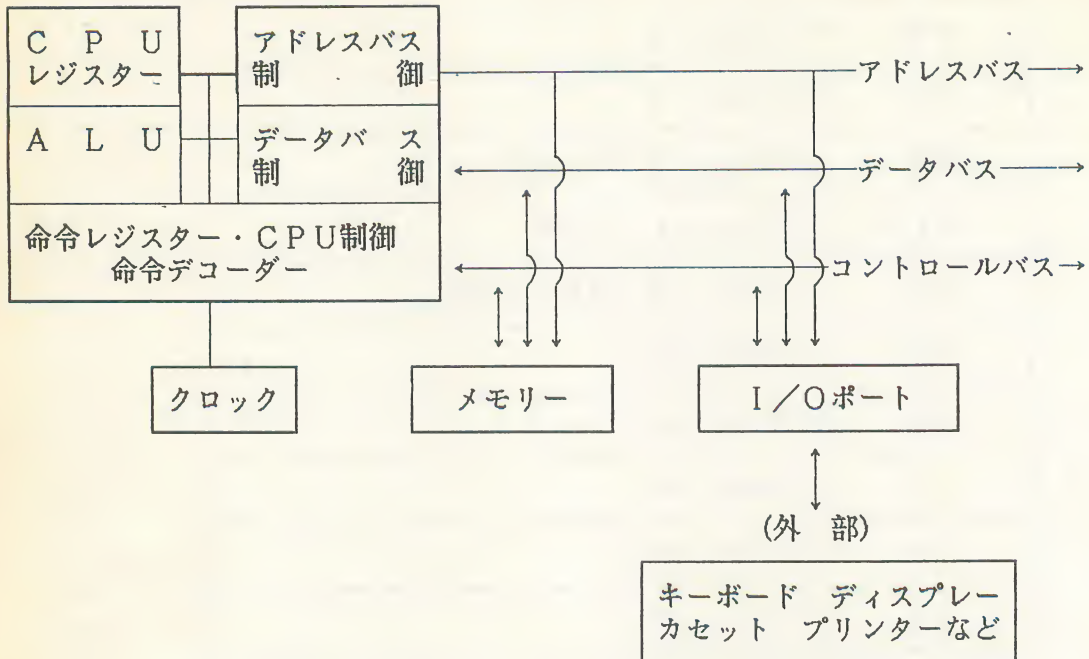
```

上例では、CTRLキーとAのキーが
押されたら、100行へとぶ。

16進コード	キ ー
08	CTRL+H DEL
09	CTRL+I
0A	CTRL+J
0B	CTRL+K HOME
0C	CTRL+L CLR
0D	CTRL+M RETURN
0E	CTRL+N
0F	CTRL+O
10	CTRL+P
11	CTRL+Q
12	CTRL+R
13	CTRL+S
14	CTRL+T
15	CTRL+U
16	CTRL+V
17	CTRL+W
18	CTRL+X
19	CTRL+Y
1A	CTRL+Z
1C	→
1D	←
1E	↑
1F	↓

CPUブロック図

CPU (Z80相当)



- CPU.....算術演算，論理演算，各種判断，制御などを行う。CPUは、時計のきざみ（クロックパルス）を処理単位として働く。
- メモリ.....プログラムやデータの記憶を1バイト（8ビット）単位とする。メモリには0～65535（&H0000～&HFFFF）まで2バイトで表されるアドレス（番地）がついている。
- I/Oポート.....CPUと外部との仲介役をする。いわばデータの出入り口。
- アドレスバス.....アドレス信号の伝送部。A0～A15まで16本の線（16ビット）の束。したがって、パラレル（並列的）に信号は伝送される。
- データバス.....CPU，メモリ，I/Oポート間のデータ伝送路。D0～D7まで8本の線（8ビット）の束。
- コントロールバス.....読み込み，書き出しなど，CPUがメモリやI/Oポートをどう動かすか指示する伝送路。
- 命令レジスタ.....データバスから読み込んだOPコード（オペコード，これが機械語）を書きこむ場所。ここへとりこむ動作を命令フェッチサイクルという。命令フェッチサイクルにあれば，データバス上のデータは無条件に命令レジスタに書かれてオペコードと解釈される。
- 命令デコーダ.....オペコードの解釈をする。
- CPU制御.....オペコードの内容や外部からの制御信号によって，CPUの各ブロックをどんな順序で，どう動かすかを判定して実行する。
- ALU.....Arithmetic Logical Unit，演算論理機構。ここで計算や論理演算をする。
- CPUレジスタ.....プログラム実行中のデータの一時記憶場所。

CPUレジスター

主レジスター		補助レジスター		専用レジスター	
A	F	A'	F'	I	R
B	C	B'	C'	I X	
D	E	D'	E'	I Y	
H	L	H'	L'	S P	
				P C	

8ビット 16ビット

- Aレジスター……アキュムレーターという。演算およびデータの読み書きの中継点。
- Bレジスター……①データの一時記憶 ②減数カウンタとしても使える。
- Cレジスター……①データの一時記憶 ②I/Oポート指定レジスターとしても使う。B, Cレジスターはペア（一対）にしてBCレジスターとして使える。
- D, Eレジスター……データの一時記憶場所。DEペアでも使える。
- H, Lレジスター……D, Eと同じ。HLペアのアドレス指定命令は豊富にある。
- Fレジスター……演算結果に応じて、変化するフラグ（ビットが1か0かで示す）をたてる。

7	6	5	4	3	2	1	0 (ビット位置)
S	Z		H		P/V	N	C

Fレジスターの中身

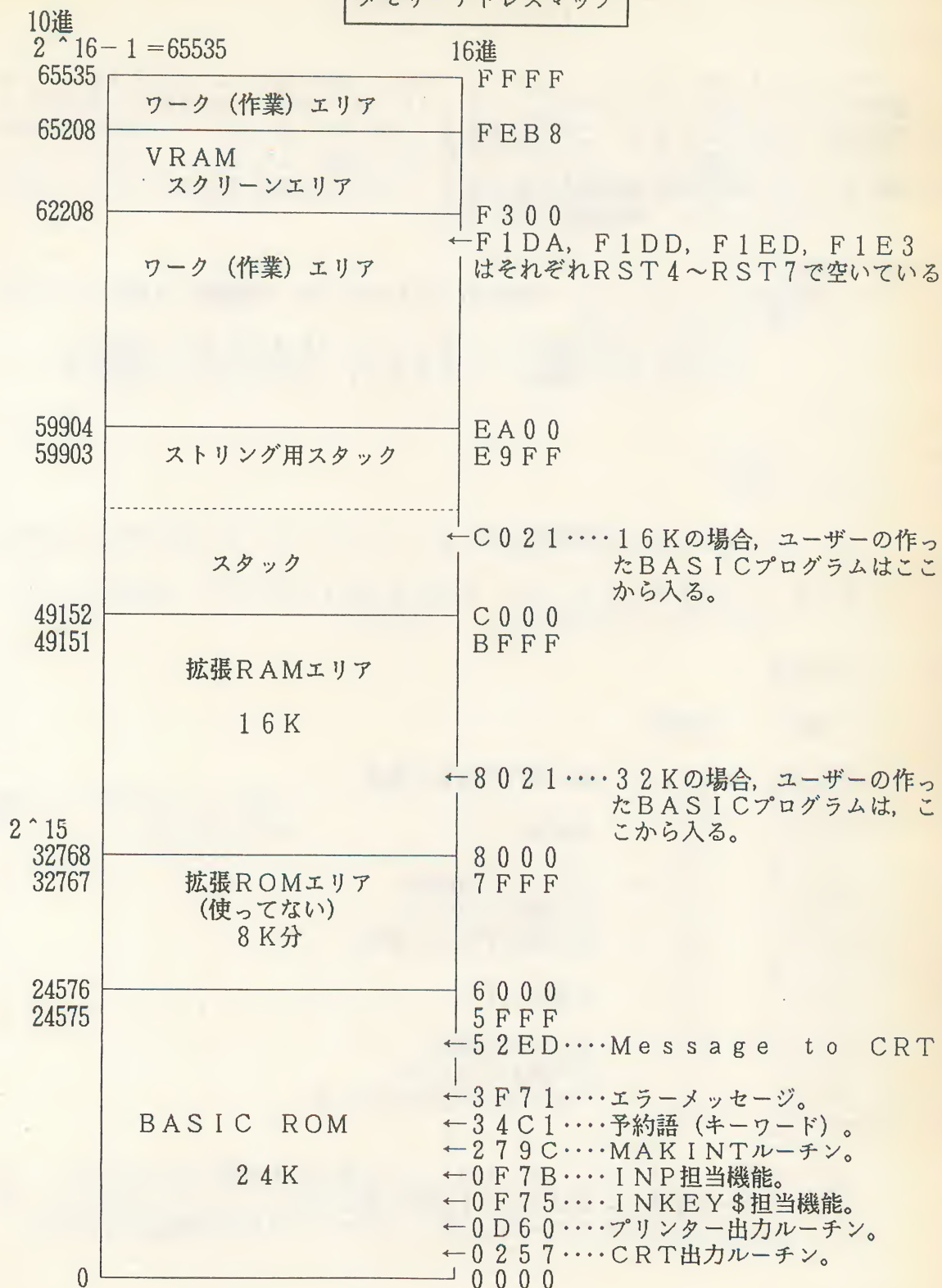
- S……サインフラグ。結果が正か負かを記憶する。
- Z……ゼロフラグ。
1……結果がゼロ。
0……結果がゼロでない。
- H……ハーフキャリーフラグ。演算の結果、3ビット目から4ビット目にキャリー（桁上がり）があったかどうか記憶。
- P/V……パリティ／オーバーフローフラグ。結果1が偶数個あるか奇数個あるかを記憶する。また、結果がオーバーフローしたかどうかを記憶。
- N……加減算フラグ。演算が加算か減算かを記憶。
- C……キャリーフラグ。CYとも書く。演算の結果、キャリー（桁上がり）が生じたかどうかを記憶。

- 補助レジスター.....この内容を直接操作する命令群はない。主レジスターと内容を交換して使う。
- Iレジスター.....Z80の割り込みモード2のときに使う。
- Rレジスター.....外部にダイナミックRAMを接続した場合、記憶が消えないようにリフレッシュするときに使う。
- IX, IYレジスター...インデックスレジスターとして使う。データのテーブル(表)を操作するときに便利。
- SPレジスター.....スタックポインター。メモリーのアドレスを意識せずにレジスターからあるメモリーにデータを出し入れできる。SPで指定するメモリーをスタックという。そのアタマを指す。プログラムであらかじめスタック領域を定め、その最高位のアドレス+1をSPに入れておく必要がある。
- PCレジスター.....プログラムカウンタ。メモリーのプログラム領域から、命令を読み出すと+1され、次の命令を順次に読み出す。

メモリーの種類

- ROM.....Read Only Memoryの略。読み出し専用のメモリー。書き込むことはできない。PC-8001の場合は24KバイトのROMがあり、そこにN-BASICが入っている。32Kバイトまで拡張できる。
- PROM.....Programmable Read Only Memoryの略。読み出し専用のメモリーだが、PROMライターという装置を使えば、プログラムを書き込むことができる。
- RAM.....Random Access Memoryの略。読み書き両方ともできるメモリー。電源を切るとメモリーの内容も消えてしまう。PC-8001の場合は、16Kバイトあり、32Kバイトまで拡張できる。

メモリーアドレスマップ



BASICの内部型

N-BASICは、インタープリターといわれる言語に属する。人間の言葉に近い言語でプログラムを書いて入力すると、トランスレーターが働いて内部表現（内部型）を作り出す。実行するときは、この内部表現をインタープリターが一つ一つ解釈しながら、コンピュータを動かす。このため、コンパイラ言語（COBOL, FORTRAN等）のように大容量の記憶装置を必要とせず、1行1行対話する形でプログラムを作ることができる。しかし、処理速度は遅くなる。

① 内部型

内部型はトランスレーターが作り出したものなので、機械語とは異なる。BASICで作ったプログラムは、

RAM 16 Kの場合……49185 (&HC021) 番地から
RAM 32 Kの場合……32801 (&H8021) 番地から

入る。

```
10 BEEP
20 END
```

といったプログラムの内部型を出すには、ダイレクトモードで以下のようにする。

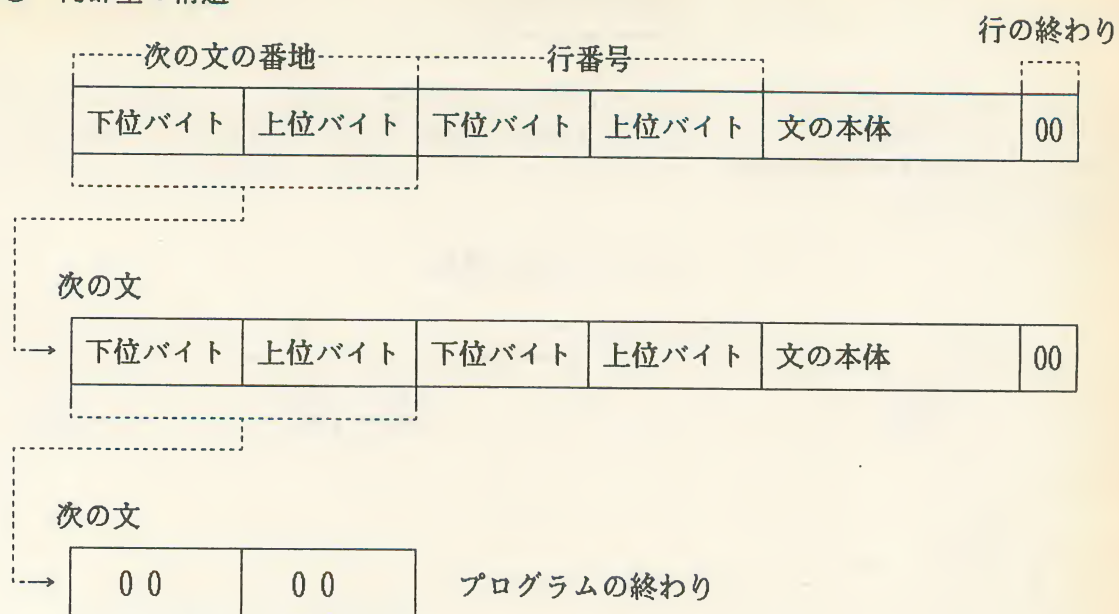
```
FOR J=&H8021 TO &H802E:PRINT HEX$(J);
" ";HEX$(PEEK(J)):NEXT
```

出力結果

番地	内部型		
8021	27	次に実行する文の番地	上位バイトと下位バイトが逆に入っている
8022	80		
8023	0A	文番号=10	
8024	00		
8025	B2	→ BEEPの内部型	
8026	00	→ 文の終わりのしるし	
8027	2D	次に実行する文の番地	
8028	80		
8029	14	文番号=20	
802A	00		
802B	81	→ ENDの内部型	
802C	00	→ 文の終わりのしるし	
802D	00	プログラムの終わりのしるし	
802E	00		

実行する際には、コンピュータがB2（BEEPの内部型）をみつけると、インタープリターが解釈して&H0D41番地へ飛んで行く。ここからBEEP, BEEP1, BEEP0のサブルーチンが入っていて、BEEPの処理が行われる。

② 内部型の構造

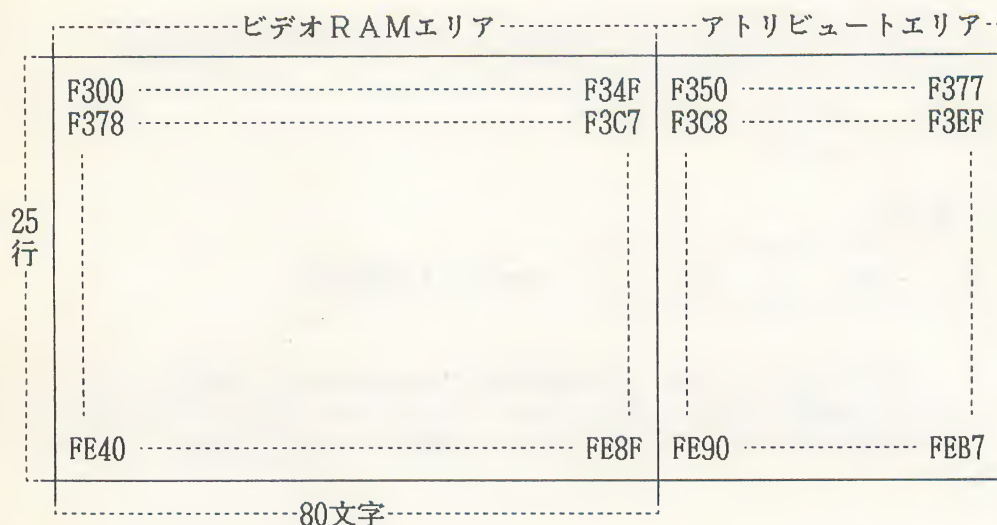


1行のプログラムは、文の本体の長さにかかわらず先頭に2バイト、末尾に1バイト必要となる。このことから、マルチステートメントにしたほうがプログラムの場所がすくなくてすむ。しかし、当然のことながら読みにくいものとなる。

ビデオRAM

PC-8001のメイン・メモリーのF300～F7B7番地は、ビデオRAM（VRAM）として使われている。ビデオRAMは、画面に表示する内容が入るビデオRAMエリアと表示する内容の属性（ファンクションコード、色）などが入るアトリビュートエリアからなる。

ビデオRAMの構成



① ビデオRAMエリア

画面上に表示される文字コード1バイトが、画面上の位置に対応した場所に入る。40文字モードの時は、偶数番地が使用され、奇数番地の内容は表示されない。カラーモードの時は、各行の先頭1番地を使用できない。20行の指定がされた時は、最初から20行分だけが使用される。

点（ドット）を表示するときは1文字を横2つ、縦4つに分割した大きさが1つの点になる。画面全体に表示できる点の数は、横〔2×（1行の文字数）〕×縦〔4×（行数）〕となる。点が表示されたとき、ビデオRAMエリアの対応する番地には、以下のコードが入る。

1文字分		2つに分割		
4 つ に 分 割	01	10	←点1つの大きさ	
	02	20		
	04	40		
	08	80		

コードは16進数で、点の位置に対応する。
01と10の位置に点があれば11がビデオRAMエリアに入る。

② アトリビュートエリア

ビデオRAMエリアと対応し、画面表示の属性が入っている。属性に変化があるとここに記録される。キャラクターモードで1行に多数の点（ドット）を表示したとき、正しく出ないことがある。これは、基本属性がキャラクターにもかかわらず点（グラフィック）を表示したことにより、アトリビュートエリアが足りなくなってしまったためである。グラフィックモードにすれば解消する。1行に20回以上のアトリビュート変更があると正しく表示されない。

*アトリビュートエリアの構成

アトリビュートエリアは、次の2バイトを1組として扱われる。

アトリビュート 変更点の位置	アトリビュート コード
-------------------	----------------

未使用のとき、アトリビュート変更点の位置には50（16進数）が入る。

*アトリビュートコードの値（16進数）

ファンクション コード	コンソールモード			
	白 黒		カラー	
	キャラクター	グラフィック	キャラクター	グラフィック
0	00	80	08	18
1	01	81	28	38
2	02	82	48	58
3	03	83	68	78
4	04	84	88	98
5	05	85	A8	B8
6	06	86	C8	D8
7	07	87	E8	F8

(例) キャラクターモードでPSET（0, 0）とするとアトリビュートエリアの先頭（F350, F351）は

00	80
----	----

となる。

*LINE文でファンクションコードを指定した場合

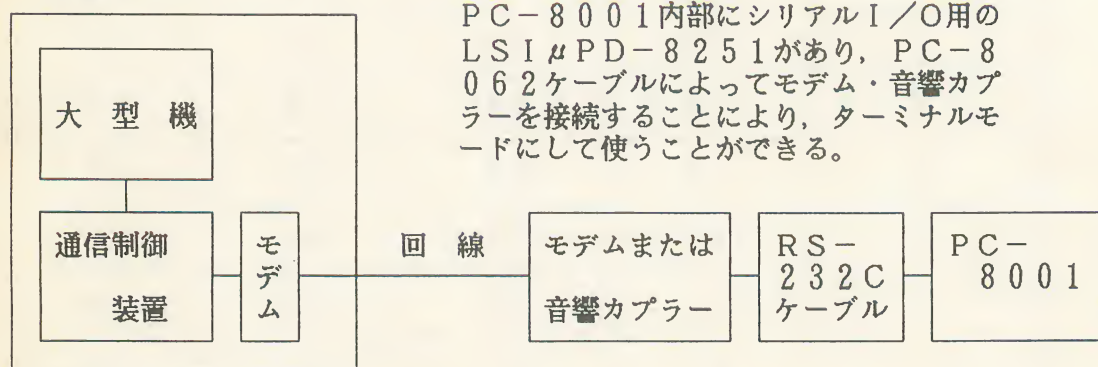
カラーモードで使用したとき、アトリビュートエリアの各行の先頭2バイトは使用されない。LINE<画面の行>、<ファンクションコード>で行単位のファンクションコードを指定した場合、この場所に以下のようなアトリビュートコード(16進数)が入る。

ファンクションコード	アトリビュートコードの値
0	00
1	02
2	04
3	06

ターミナルモード

PC-8001は、他のコンピュータのターミナルとしても使えるようにターミナルモードを持っている。

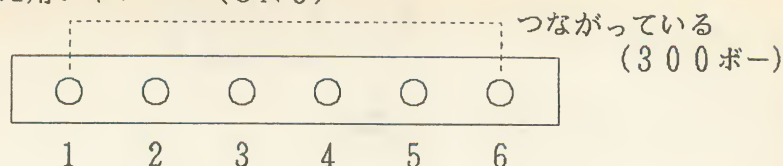
計算センター



PC-8001内部にシリアルI/O用のLSIμPD-8251があり、PC-8062ケーブルによってモデム・音響カプラーを接続することにより、ターミナルモードにして使うことができる。

計算センターによって通信方法はまちまちであるので、TERMコマンドのパラメーターで調整する。なお、基準周波数はPC内部の基板のジャンパー線を切り替えて設定する。CN8(ボーレート切り替え用)ジャンパーは300ボーに設定されている。

*ボーレート切り替え用ジャンパー (CN8)



ジャンパー 接 続	ボ ー レ ー ト	
	×16モード	×64モード
1-2	4800	1200
1-3	2400	600
1-4	1200	300
1-5	600	150
1-6	300	75

*モデム

コンピュータから出力された信号は、通常、直流信号のため、遠方へ伝送すると途中で波形が崩れてしまい正しく伝えることができない。そこで、モデム (変復調装置) によって交流信号に変換して電話回線などを通し伝送し、受け側で再びモデムで直流信号に変換して使う。

*RS-232Cインタフェース

RS-232CはEIA (米国電子工業会) が決めたコンピュータとモデムとの間のインタフェース (接続手法・装置) 規格のこと。RS-232Cインタフェース規格のものであれば、たがいに接続できる。

*IEEE-488インタフェース

計測器と計測器、コンピュータと計測器を接続する場合の接続インタフェース規格のこと。IEEE-488インタフェースを持った計測器などは、PCファミリーの拡張ユニットPC-8011を通してPC-8001のN-BASICで制御できる。

*USART

Universal Synchronous/Asynchronous Receiver/Transmitterの略。汎用同期/非同期送受信器。

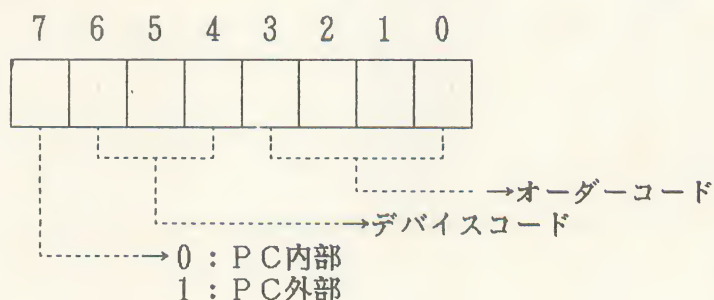
PCの内部では、データを8ビットの平行 (並列) で扱う。また、コンピュータ間のデータ伝送にはシリアル (直列) 伝送を使う。USARTは平行データとシリアルデータの仲立ちをして、相互に変換をする機能を持つLSIである。

PC-8001では、μPD8251というLSIが使われていて、モデムの制御、パリティビット (伝送チェック用) の生成やチェックなどを自動的にする。

I/O マップ

PC-8001のI/Oアドレス構成

I/Oアドレスは、1バイト（8ビット）で表されるので、256の番地（0～255）がある。PCではビット7をPC本体の外部、内部の区別に使い、ビット4、5、6を接続先のデバイス（装置）の区別に使っている。残りのビット0、1、2、3は、それぞれのデバイスにより使われかたがちがう。



内部 外部	デバイス コード	アドレス (16進)	インタフェース先	備 考
0 PC 内部	000	00～0F	(IN) キーボード00～0A	キーボードマップをみよ
	001	10～1F	(OUT) プリンター、タイマー	
	010	20～2F	(IN, OUT) シリアルチャンネル USART8251	
	011	30～3F	CPUシステムコントロ ール	
	100	40～4F	(IN) プリンター、BUSY ACK (OUT) プリンター、BEEP	
	101	50～5F	CRTコントロール	
	110	60～6F	DMAコントロール	
	111	70～7F	未使用	

I/O マップ

内部 外部	デバイス コード	アドレス	インタフェース先	備 考
1 PC 外部	000 001 010	80~AF	拡張用 I/O アドレス ユーザーに開放されてい る。	
	011	B0~BF	汎用パラレル I/O ポー ト	B0...8ビット入力ポ ート 正論理 B1...8ビット出力ポ ート 正論理 B2...4ビット入力ポ ート データバスの下位 4ビットを読み込む B3...4ビット出力ポ ート 負論理 PC80 01のリセットによりハ イレベルになる
	100	C0~CF	RS232Cインタフェ ース	C1...チャンネル1 C2...チャンネル2
	101	D0~DF	IEEE-488インタ フェース	D0...データ出力 D1...データ入力 D2...制御信号出力 D3...8255コント ロール D8...制御信号入力
	110	E0~EF	PC-8011システム コントロール	E0...モード0 E1...モード1 E2...モード2 E3...モード3 E4...RS232Cチ ャンネル使用初期設定
	111	F0~FF	フロッピーディスクコン trol	

キーボードマップ

データ・バス		D 0	D 1	D 2	D 3	D 4	D 5	D 6	D 7
ポ ー ト 番 号	0 0	0	1	2	3	4	5	6	7
	0 1	8	9	*	+	=	,	.	RETU RN
	0 2	@ `	A チ	B コ	C ソ	D シ	E イ	F ハ	G キ
	0 3	H ク	I ニ	J マ	K ノ	L リ	M モ	N ミ	O ラ
	0 4	P セ	Q タ	R ス	S ト	T カ	U ナ	V ヒ	W テ
	0 5	X サ	Y ン	Z ツ	[(\ ¥])	^ へ	_ =
	0 6	0 ワ	1 ヌ	2 フ	3 ア	4 ウ	5 エ	6 オ	7 ヤ
	0 7	(8 ュ) 9 ヨ	* :	+ ;	< ,	> .	? /	- _
	0 8	HOME CLR	↓ ↑	← →	INS DEL	GRAP H	カナ	SHIF T	CTRL
	0 9	STOP	f.1	f.2	f.3	f.4	f.5	SPAC E	ESC
		254	253	251	247	239	223	191	127

データバスの内容, 何も押さなければ255。

キーボードマップ

(グラフィックモード)

		データ・バス							
		D 0	D 1	D 2	D 3	D 4	D 5	D 6	D 7
ポ ー ト 番 号	0 0	┌	┐	+	+	⌂	≠	≠	┐
	0 1	┐	┐	—	=		┐	┐	
	0 2	■	◡	■	—	◡	◡	◡	●
	0 3	○	♠	◆	♣	■	■	■	♥
	0 4	■	◡	◡	◡	/	×	—	◡
	0 5	—	／	—		円		■	■
	0 6	秒					年	月	日
	0 7	時	分	—		■			
	0 8					GRAP H			
	0 9								
		254	253	251	247	239	223	191	127

(1) 関数

予約語マップ

関数は、頭にFFがつき、2バイトでできている。
SQRの内部コードはFF87

上段 予約語
下段 サブルーチン入りの番地

	8	9	A	B	C	D	E	F
0		INP 56A1	CSNG 27B3					
1	LEFT\$ 54F9	POS 5079	CDBL 27DF					
2	RIGHT\$ 5529	LEN 548C	FIX 282C					
3	MID\$ 5532	STR\$ 527A	CVI F0E1					
4	SGN 2686	VAL 5553	CVS F0E4					
5	INT 283F	ASC 5498	CVD F0E7					
6	ABS 2671	CHR\$ 54A8	DSKF F159					
7	SQR 31A1	PEEK 5911	EOF F0EA					
8	RND 3283	SPACE\$ 54DF	LOC F0ED					
9	SIN 32FC	OCT\$ 5270	LOF F0F0					
A	LOG 2503	HEX\$ 5275	FPOS F168					
B	EXP 31F3	LPOS 5074	MKI\$ F0F3					
C	COS 32F6	PORT 20F9	MKS\$ F0F6				IEEE	
D	TAN 335D	DEC F123	MKD\$ F0F9					
E	ATN 3372	BCD\$ F120						
F	FRE 5051	CINT 277F						

(2) コマンド、ステートメント等

予約語マップ

上段 予約語
下段 サブルーチン入り口の番地

	8	9	A	B	C	D	E	F
0		STOP 4 3 2 A	WIDTH 0 8 4 3	PRESET 0 7 0 5	MAT F 1 1 1	LSET F 1 4 4	ERR	> (注)
1	END 4 3 2 F	PRINT 4 7 4 2	ELSE 4 5 C 0	PSET 0 6 B 8	LISTEN F 1 0 E	RSET F 1 4 7	STRING \$	= (注)
2	FOR 4 1 5 9	CLEAR 4 4 E 8	TRON 4 3 9 6	BEEP 0 D 4 1	DSK0 \$ F 1 2 C	SAVE F 1 4 A	USING	< (注)
3	NEXT 4 A 0 8	LIST 5 7 0 C	TROFF 4 3 9 7	FORMAT F 1 2 F	REMOVE F 1 5 0	LFILES F 1 2 6	INSTR	+ (注)
4	DATA 4 5 B E	NEW 3 D E 0	SWAP 4 3 9 C	KEY 1 3 4 3	MOUNT F 1 5 3	INIT 2 2 6 2	'	- (注)
5	INPUT 4 8 D A	ON 4 6 4 2	ERASE 4 3 D F	COLOR 0 9 5 1	OPEN F 0 F F	LOCATE 0 7 9 2	VARPTR	* (注)
6	DIM 4 E 3 7	WAIT 5 6 B 3	ERROR 4 6 C 4	TERM 0 D B 8	FIELD F 1 0 2		CSRLIN	/ (注)
7	READ 4 9 3 9	DEF 5 0 C C	RESUME 4 6 8 C	MON 0 D 5 D	GET 1 8 8 6	TO	ATTR \$	^ (注)
8	LET 4 5 D E	POKE 5 9 1 8	DELETE 5 8 D 9	CMD F 0 F C	PUT 1 8 9 1	THEN	DSK1 \$	AND
9	GOTO 4 5 6 D	CONT 4 3 8 3	AUTO 4 6 C F	MOTOR 0 D A 1	SET F 1 5 6	TAB (INKEY \$	OR
A	RUN 4 5 3 D	CSAVE 1 E C 0	RENUM 5 A E D	POLL F 1 1 4	CLOSE F 1 3 5	STEP	TIME \$	XOR
B	IF 4 7 0 2	CLOAD 1 F 1 0	DEFSTR 4 4 5 B	RBYTE F 1 1 A	LOAD F 1 3 8	USR	DATE \$	EQV
C	RESTORE 4 3 0 2	OUT 5 6 A D	DEFINT 4 4 5 E	WBYTE F 1 1 7	MERGE F 1 3 B	FN		IMP
D	GOSUB 4 5 5 5	LPRINT 4 7 3 A	DEFSNG 4 4 6 1	ISSET F 1 0 5	FILES F 1 4 D	SPC (SRQ	MOD
E	RETURN 4 5 A 3	LLIST 5 7 0 7	DEFDBL 4 4 6 4	IRESET F 1 0 8	NAME F 1 3 E	NOT	STATUS	¥ (注)
F	REM 4 5 C 0	CONSOLE 0 8 8 4	LINE 4 8 7 7	TALK F 1 0 B	KILL F 1 4 1	ERL	POINT	

(注) 演算子としては、この内部コードになる。

ELSEは頭に1バイトつき、3 A A 1の内部コードとなる。

アポストロフ (') は頭に2バイトつき、3 A 8 F E 4の内部コードとなる。

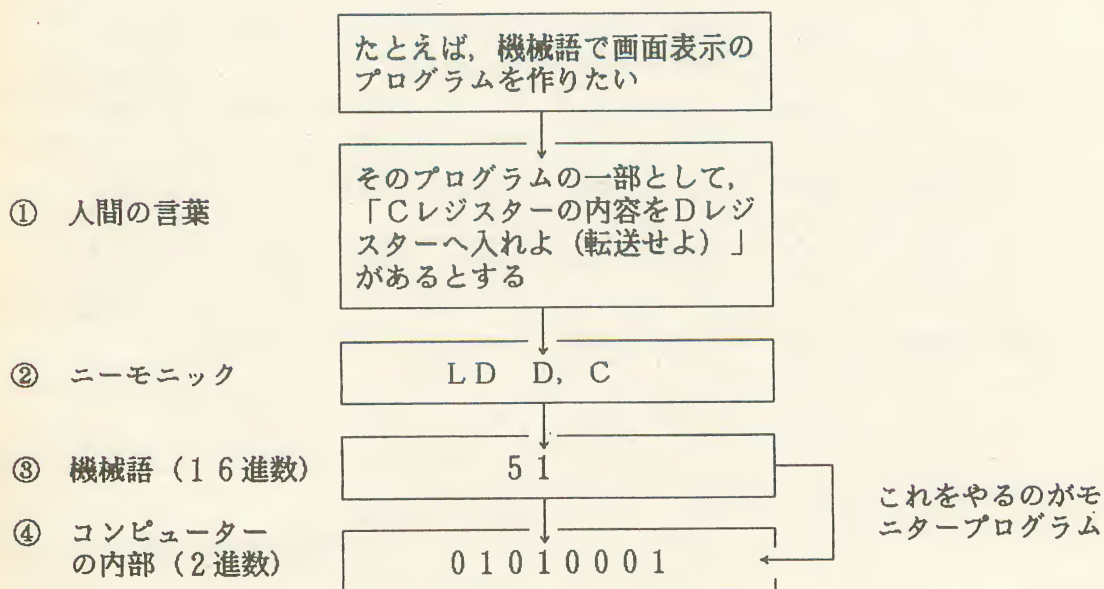
機械語プログラムの作り方

- (1) モニタープログラムは、PC内部に記憶されていて、16進数を使って人間とコンピューターとの仲立ちをしてくれる。
PC-8001には8個のモニターコマンドがある。(16ページ参照)
- (2) MON (RETキー) でモニターモードになり、画面左端にアスタリスク(*)が表示される。このアスタリスクがBASICモードのときの「OK」に相当する。
*の次にDと入力してみよ。
これでメモリーの0000番地から000F番地までの内容が画面に表示される。表示されたF3, 31 ----- は、N-BASICの中身そのものであり、このように16進数の機械語で出てくる。
- (3) コンピューターの内部では、電圧の高低を利用し(高いところを1とすれば低いところが0)これを2進法の1と0とにみたてて、すべての作業をしている。
しかし、2進数では書くのがめんどうなので、2進数4桁を16進数1桁として表示する方法がとられる。1バイト分は次のようになる。

1バイト = 2進数8桁 = 16進数2桁

モニターモードでは、16進数2桁で入力すれば、モニタープログラムが働いて、コンピューターが理解できる2進数に変換してくれる。

- (4) 人間にとっては16進数でも分かりにくい。そこでニーモニック(Mnemonic)で表現することが考えられた。ニーモニックには、「記憶する」という意味がある。



- (5) 最終的には、④の段階の2進数にしなければならないが、PCでは③の段階の16進数で入力すれば、モニタープログラムが働いて④の段階に直してくれる。
- ②のニーモニックの段階で入力して④の段階に直してくれるものをアセンブラーという。したがって、PC用のアセンブラーを作りたければ、ニーモニック→機械語変換表とそれを索引して変換してくれるプログラムが要る。逆に、機械語をニーモニックに変換してくれるプログラムを逆アセンブラーという。
- (6) ハンドアセンブルとは、アセンブラーの役目を人間が手作業でやることだから、ニーモニックで書いたものを、ニーモニック→機械語変換表をみて16進数に直すことをいう。
- (7) 機械語を入力するには、まず、モニターモードにしておく。
*S××××番地(RETキー)で××××番地から入力できる。入力した機械語の確認は*D××××番地(RETキー)で行う。
- (8) 機械語プログラムを実行するときも、最初にモニターモードにしておかねばならない。特殊な例としてDISKの場合があるが、普通は使わないので省略する。
*G××××番地(RETキー)で××××番地からの機械語プログラムが動き出す。
- (9) 機械語プログラムがHALT(機械語では76H)で終わると、機械自体が停止してしまうので、制御できなくなる。リセットボタンを押せばBASICモードに戻り、機械語のプログラムは残る。
要は、プログラムが終了したらMONに戻ればいいわけだから(97ページの予約語MONをみると0D5D番地。しかし、ここからさらにサブルーチンへとんで結局は5C66番地) JP 5C66H とすれば、モニターモードに戻って * が画面表示される。
BASICモードに戻りたければ、JP 0でRESETボタンを押した状態(コールドスタート)とするか、または JP 8でSTOPキー+RESETボタンを押した状態(ウォームスタート)とすればよい。
- (10) 機械語プログラムだけで、技術計算などをやろうとすると、関数やら浮動小数点アキュムレーターやらをすべて自作しなければならない。したがって、BASICモードでやったほうがいだろう。機械語の分野は、むしろ、計算の要らない画面操作などだろう。ゲームなどの画面処理速度は格段にあがる。
- (11) 命令によって機械語の長さは違う。1バイトから4バイトまでである。
- (12) 回路が簡単になるとか速度があがるとか、製造上の理由から、機械語では番地の上位と下位が「さかさま」になっている。
たとえば CALL 0E26H は CD 26 0E
- (13) 現在番地を基準に、ある番地(相対番地)へジャンプする場合、JR nを使うが、2バイト命令なのでPCカウンターが2番地先を指していることに注意。10H番地にJRがあり、20H番地へジャンプするには、 $20H - 10H - 2H = 0EH$ で、JR 0EHとなる。逆に5H番地へ行くには、 $5H - 10H - 2H = -0DH$ これを2の補数で表現するとF3H JP F3H(2進数の補数はビットを反転させて1を足す)

<機械語プログラムの例1>

*機能

キーボード上の任意のキーを押すと音が出る。STOPキーを押すとBASICモードに戻る。

*ポイント

音を出すのは、BEEPの機能を利用している。画面消去をするのは、きれいな音を出すため。(文字コードは78ページのキャラクターコード表を見よ)

また、CALLした0F75H番地はN-BASICのルーチンだが、これで押されたキーのキャラクターコードがAレジスターに入る。

番	地	機械語	ニーモニック表示	
D000	3E00	LD	A, 00H	画面消去
D002	D351	OUT	(51H), A	51HはI/Oマップ参照
D004	CD750F	CALL	0F75H	押されたキーがAレジスターに
D007	FE03	CP	03H	STOPキーが押されたら
D009	2816	JR	Z, 16H	D021番地へ
D00B	5F	LD	E, A	
D00C	0E00	LD	C, 00H	繰り返す回数を256回にする
D00E	43	LD	B, E	
D00F	3E20	LD	A, 20H	音を出しはじめる
D011	D340	OUT	(40H), A	40HはI/Oマップ参照
D013	10FE	DJNZ	FEH	キーの文字コードの値だけ待つ
D015	43	LD	B, E	
D016	3E00	LD	A, 00H	音を止める
D018	D340	OUT	(40H), A	40HはI/Oマップ参照
D01A	10FE	DJNZ	FEH	キーの文字コードの値だけ待つ
D01C	0D	DEC	C	繰り返す回数を1減らす
D01D	20EF	JR	NZ, EFH	終わりでなければD00E番地
D01F	18E3	JR	E3H	終わりならD004番地へ
D021	C30800	JP	08H	BASICに戻る(ウォームスタート)

BASICモードに戻るのではなく、モニターモードに戻る場合はD021番地以降D05B番地までに、次の機械語をつけ加えればよい。画面の初期設定をやった後でモニターに戻る。

番	地	機械語	番	地	機械語	番	地	機械語
D021	3E00		D035	3E8B		D049	3E53	
D023	D351		D037	D365		D04B	D350	
D025	3E80		D039	3ECE		D04D	3E43	
D027	D368		D03B	D350		D04F	D351	
D029	3E00		D03D	3E98		D051	3EC4	
D02B	D364		D03F	D350		D053	D368	
D02D	3EF3		D041	3E67		D055	3E20	
D02F	D364		D043	D350		D057	D351	
D031	3EB7		D045	3EDE		D059	C3665C	
D033	D365		D047	D350				

<機械語プログラムの例2>

*機能

キーボードから入力した文字を画面上に出力するとともに、プリンターにも出力する。プリンターは、バッファを持っているのですぐにはプリントしない。RETURNキーを押すか、バッファがいっぱいになるとプリントする。改行をする場合は、CTRLキーとJを押すこと。STOPキーを押すとモニターモードに戻る。

*ポイント

画面消去・キーボードのどのキーを押したか・画面出力・プリンター出力は、N-BASIC内のルーチンを利用している。

番	地	機械語	ニーモニック表示	
C000	CD5A04	CALL	045AH	PRINT CHR \$(12)と同じ
C003	CD750F	CALL	0F75H	押されたキーがAレジスターに
C006	FE03	CP	03H	STOPキーならモニター
C008	CA665C	JP	Z, 5C66H	モードに戻る
C00B	CD2B00	CALL	002BH	プリンターに出力
C00E	CD3500	CALL	0035H	画面に出力
C011	18F0	JR	F0H	

<機械語プログラムの例3>

*機能

レジスターの内容を画面に表示する。機械語プログラムのテストをするときに便利。テストをするときは、あらかじめF1E3番地から3バイトにJP E000H（機械語でC300E0）を入れておく。したがって、レジスターの内容を知りたい場所にRET 38H（機械語でFF）を入れれば、F1E3番地にとび（そこからこのプログラムのE000H番地へさらにとぶ。F1E3番地がフック番地になっている。フックとは「ひっかける」ことで中継番地）このプログラムが実行される。

*ポイント

52EDH番地の文字列出力ルーチンを利用している。任意の場所でレジスターの内容などを確認するとき、その場所をブレイクポイントという。

出力結果

AF	BC	DE	HL	
0A4A	5C5E	5C82	E000	←主レジスターの内容
AF	BC	DE	HL	
2024	E9CF	0101	D3BA	←補助レジスターの内容
IX	IY	PC+1	SP	
94F8	962E	41FA	E8D1	←専用レジスター・プログラムカウンタ・スタックポインターの内容

AFの内容は0A4AH。したがってAレジスターの内容は、0AHとなる。0AHは10進数で10、2進数で00001010。Fレジスターの内容4AHを2進数に直すと、01001010になる（83ページCPUレジスター参照）

プログラムの本体は次のようになる。

番	地	機械語	ニーモニック表示	
E000	FDE5	PUSH	IY	IX, IYをスタックに
E002	DDE5	PUSH	IX	退避
E004	CD2DE0	CALL	E02DH	主レジスタの表示
E007	08	EX	AF, AF'	レジスター交換
E008	D9	EXX		
E009	CD2DE0	CALL	E02DH	補助レジスタの表示
E00C	CDCA5F	CALL	5FCAH	画面の改行をする
E00F	215BE0	LD	HL, E05BH	IX, IYレジスター等の
E012	CDED52	CALL	52EDH	見出しを表示
E015	CDCA5F	CALL	5FCAH	画面の改行をする
E018	0603	LD	B, 03H	
E01A	E1	POP	HL	
E01B	CDC05E	CALL	5EC0H	IX, IY, PC+1の内
E01E	CDD45F	CALL	5FD4H	容を16進4桁で表示
E021	10F7	DJNZ	F7H	
E023	210000	LD	HL, 0000H	スタックポインタの値を
E026	39	ADD	HL, SP	取り出し16進4桁で表示
E027	CDC05E	CALL	5EC0H	
E02A	C3665C	JP	5C66H	モニターに戻る
E02D	E5	PUSH	HL	(ここからサブルーチン)
E02E	D5	PUSH	DE	HL, DE, BC, AFを
E02F	C5	PUSH	BC	スタックに退避
E030	F5	PUSH	AF	
E031	CDCA5F	CALL	5FCAH	画面の改行をする
E034	2149E0	LD	HL, E049H	主レジスター, 補助レジ
E037	CDED52	CALL	52EDH	ターの見出し表示
E03A	CDCA5F	CALL	5FCAH	画面の改行をする
E03D	0604	LD	B, 04H	
E03F	E1	POP	HL	HL, DE, PC, AFの
E040	CDC05E	CALL	5EC0H	内容を16進4桁で表示
E043	CDD45F	CALL	5FD4H	
E046	10F7	DJNZ	F7H	
E048	C9	RET		(サブルーチンの終わり)
E049	41462020			
E04D	20424320			主レジスター, 補助レジ
E051	20204445			ターの見出し
E055	20202048			
E059	4C00			
E05B	49582020			
E05F	20495920			IX, IY, PC+1,
E063	20205043			SPの見出し
E067	2B312053			
E06B	5000			

<機械語プログラムの例4>

*機能

図形を動かす。ドットで描かれた図形が動き続ける。

*ポイント

8文字分(横4, 縦2)の図形をドットに分解してVRAM上に描く(C06EH~C075H番地。88ページ参照)。図形の点滅はXORをとることによってする。画面の初期設定はN-BASIC内のサブルーチンWIDTH(0843H番地。97ページ予約語マップ参照), CONSOLE(0884H番地。同), COLOR(0951H番地。同)を利用している。いずれもパラメータの先頭番地をHLレジスタに入れて, CALLしてやればよい。C076H番地以降が各パラメータ。キャラクターコード(78ページ参照)で入れ, パラメータの終わりは00Hとすること。C04BHからのサブルーチンで表示と消去をしている。C064H番地に画素の先頭番地を入れておき, C066H~C06DH番地に画素の先頭番地から何番地はなれているかのテーブルを作っておく。先頭番地とテーブルから表示する番地を割り出して, 対応する画素をその番地に書き込む。機械語で画面に表示するときは処理速度が速すぎるので適度に時間をかせぐことが必要(C044Hがカラまわしのサブルーチン)。C021H, C033H番地はNOPで特別な意味はない。ニーモニックは省略する。

番 地	機械語	番 地	機械語
C000	2176C0	C044	C5
C003	CD4308	C045	0600
C006	217CC0	C047	10FE
C009	CD8408	C049	C1
C00C	2185C0	C04A	C9
C00F	CD5109	C04B	0608
C012	CD5A04	C04D	DD2166C0
C015	2100F3	C051	1600
C018	2264C0	C053	DD5E00
C01B	060A	C056	2A64C0
C01D	C5	C059	19
C01E	CD4BC0	C05A	DD7E08
C021	00	C05D	AE
C022	0600	C05E	77
C024	CD44C0	C05F	DD23
C027	10FB	C061	10EE
C029	CD4BC0	C063	C9
C02C	0600	C064	0000
C02E	CD44C0	C066	00010203
C031	10FB	C06A	78797A7B
C033	00	C06E	216CC612
C034	2A64C0	C072	48633684
C037	1600	C076	38302C32
C039	1EF4	C07A	3500
C03B	19	C07C	302C3235
C03C	2264C0	C080	2C302C30
C03F	C1	C084	00
C040	10DB	C085	302C302C
C042	18D1	C089	3100

Z 8 0 相当代表的機械語説明表

語尾のHは16進を表す

機械語 (16進)	ニーモニック	説明
5 1	LD D, C	Cレジスタの内容をDレジスタに転送 (ロード) せよ LD D, C ┌──→ソースフィールド └──→デスティネーション フィールド └──→オペコードフィールド
0 E n	LD C, n	nをCレジスタに転送せよ。nは1バイトの数値。16進なら語尾にHをつける。0 E 77Hなら、77HをCレジスタに転送せよ。2バイト命令
3 6 n	LD (HL), n	nをHLレジスタが示す番地に転送せよ。()があったら、そのレジスタが示す番地の内容
0 1 n n	LD BC, n n	LD BC, 4000Hは01 00H 40Hと上位と下位が逆になることに注意。3バイト命令。なおデータの語尾のHは、機械語には必要ない
2 A n n	LD HL, (n n)	LD HL, (6000H)は2A 00H 60H Hに6001H番地の内容が、Lに6000H番地の内容が入る
2 2 n n	LD (n n), HL	LD (1000H), HLは22 00H 10Hとなる。Hの内容を1001H番地に転送し、Lの内容を1000H番地に転送する
C 5	PUSH BC	BCの内容をスタックポインターで示す番地へ転送。BがSP (スタックポインター) の内容-1番地へ転送され、CがSPの内容-2番地へ転送される。SPは-2される。あらかじめSPにスタックの番地+1を入れておくこと

機械語 (16進)	ニーモニック	説明
C1	POP BC	PUSHの逆。SPで示された番地の内容がCに入り、SP+1で示された番地の内容がBへ入る。SPは+2される
80	ADD A, B	$A \leftarrow A + B$ Aレジスタの内容にBレジスタの内容を加え、Aレジスタの内容とする。以下、レジスタの語を省略して説明する
09	ADD HL, BC	$HL \leftarrow HL + BC$
8B	ADC A, E	$A \leftarrow A + E + CY$ CYは桁あふれのキャリーフラグ。演算の結果アキュムレータの最上位ビットが桁あふれするとキャリーフラグがたつ。2バイト以上の数の和を求めるとき下位バイトどうしの和でキャリーがでたら、この命令で上位バイトどうしの和の方へ加えてやることができる
90	SUB B	$A \leftarrow A - B$
98	SBC A, B	$A \leftarrow A - B - CY$
A0	AND B	$A \leftarrow A \text{ AND } B$
B8	CP B	A-Bをやり、結果はすべてフラグの変化だけを求める。Aは変化しない。比較命令
04	INC B	$B \leftarrow B + 1$ インクリメント命令
05	DEC B	$B \leftarrow B - 1$ デクリメント命令
DD.86n	ADD A, (ix + n)	Aの内容を10H、ixの内容を1000H、nを5Hとすると、1005H番地の内容とAの内容10Hが加算され、Aの内容となる。DDがつくとix（インデックスレジスタ）が使われる
FD86n	ADD A, (iy + n)	FDがつくとIYレジスタ

機械語 (16進)	ニーモニック	説明
EB	EX DE, HL	DE \longleftrightarrow HL。DEとHLの内容を交換する
D9	EXX	BC, DE, HL \longleftrightarrow B' C' , D' E' , H' L'
E3	EX (SP), HL	H \longleftrightarrow (SP+1) L \longleftrightarrow (SP)
DDE3	EX (SP), IX	IX (H) \longleftrightarrow (SP+1) IX (L) \longleftrightarrow (SP)
C3 nn	JP nn	特定番地へジャンプ。JP 300 0HはC3 00H30H。3バイト命令
18 n	JR n	現在番地から相対的に指定した番地へとぶ。2バイト命令でプログラムカウンタが2番地先へ進んでいることに注意する。 E000H番地からE012H番地へとびたいときは E012H-E000H-2H =10H 故に JR 10H
E9	JP (HL)	HLの内容が1234Hなら、そこへとぶ
CA nn	JP Z, nn	Z (ゼロ) フラグを見て、1ならnn番地へとぶ。NZなら0でnnへとぶ
DA nn EA nn E2 nn F2 nn FA nn	JP C, nn	CY (キャリー) フラグをみて1ならnn番地へとぶ 同様に JP PEはパリティ・イーブン (パリティが偶数なら) 同様に JP POはパリティ・オッド (パリティが奇数なら) 同様に JP Pはサイン・ポジティブ (正) 同様に JP Mはサイン・ネガティブ (負) 要するに条件が真なら指定番地へとぶ


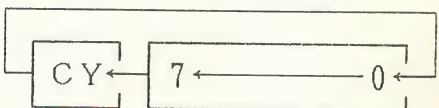

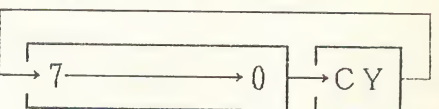
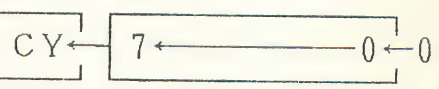
機械語 (16進)	ニーモニック	説明
10 n	DJNZ n	DEC BとJR NZとの合成。 B←B-1をやって0でなければシ ャンプ。0なら次の命令を実行。シ ャンプ番地の計算はJR nとおな じ
CD nn	CALL nn	nn番地からはじまるサブルーチン へとぶ
C4 nn	CALL NZ, nn	ゼロフラグがたっていないければ (N Z), nn番地へとぶ
C9	RET	サブルーチンの最後へ書くリターン 命令。CALLの次の命令へ帰る
C8	RET Z	Z (ゼロ) ならもどる
ED4D	RETI	基本的にはRETだが、割り込み処 理ルーチンに使われ、割り込み終了 を周辺装置に知らせる
ED45	RETN	NMI (ノンマスクابل割り込み) に対するサービスルーチンの終了
DF	RST 18H	RST (リスタート) はCALLと 同じ働きをする。この例は18Hを 4桁の0018H番地とみて、そこ がサブルーチン開始番地となる
EDA0	LDI	(HL) 番地の内容を (DE) 番地 の内容に書く。続いてHLとDEは +1され、BCは-1される。ブロッ ク転送命令
EDB0	LDIR	同上。BCが0になったらブロック 転送終了。すなわちBCはバイトカ ウンター。100バイト転送したけ ればBCに100をセットする
EDA8	LDD	(DE) ← (HL) は同じ。DE, HL, BCとも-1される
EDB8	LDDR	同上。BCが0になったら、ブロッ ク転送を止める

機械語 (16進)	ニーモニック	説明
EDA1	CPI	Aの内容と(HL)番地の内容とを比較。A, (HL)は不変。レジスタのZが $A = (HL)$ なら1, P/V が $BC-1 \neq 0$ なら1。比較後, BCは-1, HLは+1される
EDB1	CPIR	同上。A=(HL)か $BC=0$ のどちらかが成立すると命令実行終了
EDA9	CPD	CPIと同じだが, 比較後, BCは-1, HLも-1される
EDB9	CPDR	CPIRと同じ機能。BC, HLとも-1される点が違う。条件成立までCPDと同じ操作を繰り返す
27	DAA	2進の加減算命令で, BCD(2進化10進)表現の数値間の計算をした場合, BCDコードに直す 2進計算なら $27H + 58H = 7FH$ BCD計算なら $27H + 58H = 85H$ Aの内容が7FHとなっているのを85Hに直す。ADDやADC, SUB, SBCと組み合わせて使う 例 ADD A, C DAA
2F	CPL	$A \leftarrow \text{NOT } A$ Aの内容の0を1に, 1を0にする
ED44	NEG	$A \leftarrow 0 - A$ Aの内容を正から負, 負から正にする。すなわちAの内容を2の補数にする
3F	CCF	キャリーフラグが反転する
37	SCF	セットキャリーフラグ。キャリーフラグを1にする
00	NOP	ノーオペレーション。なにもしないで次の命令へ行く

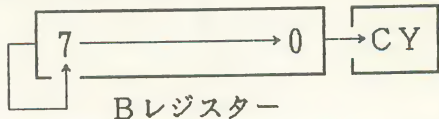
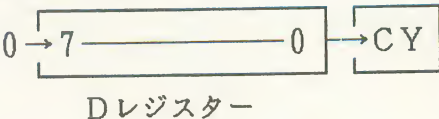
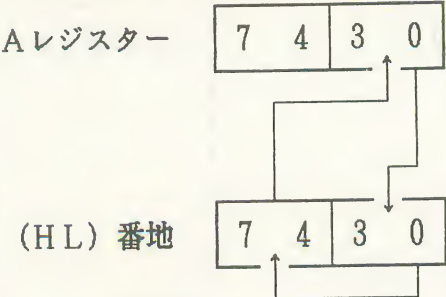
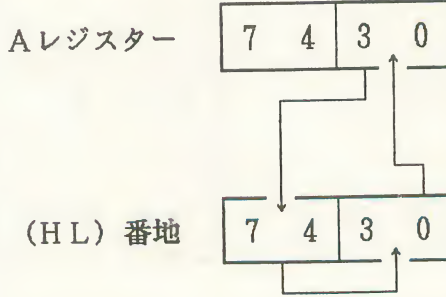
語尾のHは16進を表す

機械語 (16進)	ニーモニック	説明
76	HALT	CPU待機。Z80のプログラムの実行を停止する
F3	DI	ディスエーブル・インタラプト。割り込みを禁止する
FB	EI	イネーブル・インタラプト。割り込みを可能にする
ED46	IM 0	割り込みモード0。割り込みを受けつけると、データバスから割り込み命令を入れて、割り込み処理をさせる。すなわち、コールする番地は周辺デバイスからCALLやRSTを入れておこなう
ED56	IM 1	同モード1。割り込みを受けつけると、0038H番地を無条件にコール。割り込み処理プログラムは0038H番地から入れておかねばならない
ED5E	IM 2	同モード2。割り込みを受けつけると自動的にコール命令を実行する コールする番地は、 上位8ビット・・・Iレジスターから 下位8ビット・・・データバスから ただし、最下位ビットは0として間接コール IM 0～IM 1は、INT割り込みのやりかたをきめる 割り込みはEIで可能になるが、割り込みが起こるとDIの状態になるので、EIを割り込みルーチンのRETIの前にいれEI状態に戻す NMI (ノンマスカブル・インタラプト) の場合は、割り込みを受けつけると無条件に0066H番地からはじまるプログラムをコールする これは、RETNで戻れる
DB n	IN A, (n)	入力ポートnを指定してやると、その内容がAに入る

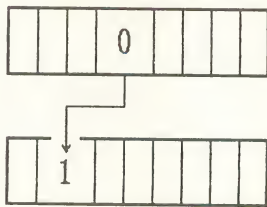
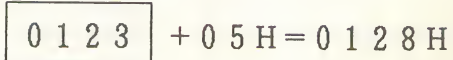
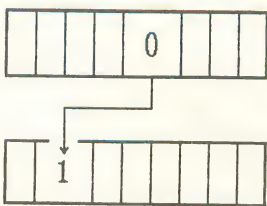
機械語 (16進)	ニーモニック	説明
D3 n	OUT (n), A	指定した出力ポートに、Aの内容を出力する
ED50	IN D, (C)	Cレジスタの内容を、入力ポートアドレスとして、その入力ポートからの入力データをDレジスタに入れる
ED41	OUT (C), B	Cレジスタの内容を、出力ポートアドレスとして、Bレジスタの内容を出力する
EDA2	INI	Cレジスタの内容を入力ポートアドレスとして、入力データを、HLが示す番地へ書く。HLは+1されBは-1される
EDA3	OUTI	INIの逆。HLは+1され、Bは-1される すべてのOUT命令で、アドレスバスの下位8ビット (A7~A0) はCの内容、上位8ビット (A15~A8) にはBの内容が出力される
EDB2	INIR	INI命令をBレジスタの内容だけ (B=0になるまで) くりかえす
EDB3	OTIR	OUTI命令を同上くりかえす
EDAA	IND	INIとちがう点は、HLが-1される
EDAB	OUTD	OUTIとちがう点は、HLが-1される
EDBA	INDR	INIRではHLは+1される INDRではHLが-1される
EDBB	OTDR	OTIRではHLは+1される OTDRではHLが-1される

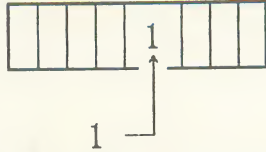
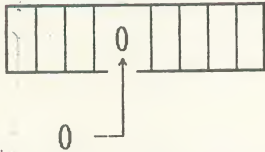
機械語 (16進)	ニーモニック	説明
07	RLCA	<p>左ローテート・サーキュラー・アキュムレーター 1ビット左回転</p>  <p>Aレジスター</p>
17	RLA	<p>左ローテート・アキュムレーター CYを含み1ビット左回転</p>  <p>Aレジスター</p>
0F	RRCA	<p>右ローテート・サーキュラー・アキュムレーター 1ビット右回転</p>  <p>Aレジスター</p>
1F	RRA	<p>右ローテート・アキュムレーター CYを含み1ビット右回転</p>  <p>Aレジスター</p> <p>RLCB, RR(HL), RLC(IX+n)などは、レジスターが異なったり、番地の内容が対象になるだけで、以上に準ずる</p>
CB23	SLA E	<p>左シフト命令</p>  <p>Eレジスター</p>

語尾のHは16進を表す

機械語 (16進)	ニーモニック	説明
CB28	SRA B	右シフト命令 
CB3A	SRL D	右シフト命令 
ED6F	RLD	左ローテート・ディジット・アキュムレーター Aの上位4ビット不変 Aレジスター 
ED67	RRD	右ローテート・ディジット・アキュムレーター Aの上位4ビット不変 Aレジスター 

語尾のHは16進を表す

機械語 (16進)	ニーモニック	説明
CB 6 7	BIT 4, A	<p>Aレジスターの4ビット目 (5桁目になる) が1であればFレジスターのZに0 (NO) を書く。0であればZに1 (YES) を書く テストビット命令</p> <p>Aレジスター 7 6 5 4 3 2 1 0</p>  <p>S Z H PV NC Fレジスター</p> <p>例 BIT 4, A JP Z, E 0 0 0 H</p> <p>Aレジスターの4ビット目が0ならE 0 0 0 H番地へとぶ</p>
DDCB n 5 E	BIT 3, (iX+n)	<p>同上, 4バイト命令になる。iXを0 1 2 3 Hとし, nを0 5 Hとすると, BIT 3, (iX+0 5 H)でDDCB 0 5 H 5 Eとなる</p> <p>i X</p>  <p>0 1 2 8 H 7 6 5 4 3 2 1 0</p>  <p>S Z H PV NC Fレジスター</p> <p>FDがつくとiYになるのは同じ</p>

機械語 (16進)	ニーモニック	説明
CBD9	SET 3, C	<p>セットビット。Cレジスタの3ビット目を1にする</p> <p>Cレジスタ</p> <p>7 6 5 4 3 2 1 0</p> 
CBA0	RES 4, B	<p>リセットビット。Bレジスタの4ビット目を0にする</p> <p>Bレジスタ</p> <p>7 6 5 4 3 2 1 0</p> 
<p>(注) 1. nは1バイトの数 2. nnは2バイトの数。機械語では上位バイト, 下位バイトが逆になる 3. () がつく, () の中の数が番地として使われる 4. i Xを含む命令は頭にDDがつく 5. i Yを含む命令は頭にFDがつく 6. DDCB, FDCBの4バイト命令は, 3バイト目にnが入る 7. (i X+n) の場合, nは-128 (80H) ~127 (7FH) 8. AFとA' F' の交換は, EX AF, A' F' とは書かず, EX A F, AF' と書く</p>		

機械語	ニーモニック	機械語	ニーモニック
00	NOP	16	LD D, n
01	LD BC, nn	17	RLA
02	LD (BC), A	18	JR n
03	INC BC	19	ADD HL, DE
04	INC B	1A	LD A, (DE)
05	DEC B	1B	DEC DE
06	LD B, n	1C	INC E
07	RLCA	1D	DEC E
08	EX AF, AF'	1E	LD E, n
09	ADD HL, BC	1F	RRA
0A	LD A, (BC)	20	JR NZ, n
0B	DEC BC	21	LD HL, nn
0C	INC C	22	LD (nn), HL
0D	DEC C	23	INC HL
0E	LD C, n	24	INC H
0F	RRCA	25	DEC H
10	DJNZ n	26	LD H, n
11	LD DE, nn	27	DAA
12	LD (DE), A	28	JR Z, n
13	INC DE	29	ADD HL, HL
14	INC D	2A	LD HL, (nn)
15	DEC D	2B	DEC HL

機械語	ニーモニック	機械語	ニーモニック
2C	INC L	42	LD B, D
2D	DEC L	43	LD B, E
2E	LD L, n	44	LD B, H
2F	CPL	45	LD B, L
30	JR NC, n	46	LD B, (HL)
31	LD SP, nn	47	LD B, A
32	LD (nn), A	48	LD C, B
33	INC SP	49	LD C, C
34	INC (HL)	4A	LD C, D
35	DEC (HL)	4B	LD C, E
36	LD (HL), n	4C	LD C, H
37	SCF	4D	LD C, L
38	JR C, n	4E	LD C, (HL)
39	ADD HL, SP	4F	LD C, A
3A	LD A, (nn)	50	LD D, B
3B	DEC SP	51	LD D, C
3C	INC A	52	LD D, D
3D	DEC A	53	LD D, E
3E	LD A, n	54	LD D, H
3F	CCF	55	LD D, L
40	LD B, B	56	LD D, (HL)
41	LD B, C	57	LD D, A

HL ← mm

機械語	ニーモニック	機械語	ニーモニック
58	LD E, B	6E	LD L, (HL)
59	LD E, C	6F	LD L, A
5A	LD E, D	70	LD (HL), B
5B	LD E, E	71	LD (HL), C
5C	LD E, H	72	LD (HL), D
5D	LD E, L	73	LD (HL), E
5E	LD E, (HL)	74	LD (HL), H
5F	LD E, A	75	LD (HL), L
60	LD H, B	76	HALT
61	LD H, C	77	LD (HL), A
62	LD H, D	78	LD A, B
63	LD H, E	79	LD A, C
64	LD H, H	7A	LD A, D
65	LD H, L	7B	LD A, E
66	LD H, (HL)	7C	LD A, H
67	LD H, A	7D	LD A, L
68	LD L, B	7E	LD A, (HL)
69	LD L, C	7F	LD A, A
6A	LD L, D	80	ADD A, B
6B	LD L, E	81	ADD A, C
6C	LD L, H	82	ADD A, D
6D	LD L, L	83	ADD A, E

機械語	ニーモニック	機械語	ニーモニック
84	ADD A, H	9A	SBC A, D
85	ADD A, L	9B	SBC A, E
86	ADD A, (HL)	9C	SBC A, H
87	ADD A, A	9D	SBC A, L
88	ADC A, B	9E	SBC A, (HL)
89	ADC A, C	9F	SBC A, A
8A	ADC A, D	A0	AND B
8B	ADC A, E	A1	AND C
8C	ADC A, H	A2	AND D
8D	ADC A, L	A3	AND E
8E	ADC A, (HL)	A4	AND H
8F	ADC A, A	A5	AND L
90	SUB B	A6	AND (HL)
91	SUB C	A7	AND A
92	SUB D	A8	XOR B
93	SUB E	A9	XOR C
94	SUB H	AA	XOR D
95	SUB L	AB	XOR E
96	SUB (HL)	AC	XOR H
97	SUB A	AD	XOR L
98	SBC A, B	AE	XOR (HL)
99	SBC A, C	AF	XOR A

機械語→ニーモニック

(B 0 ~ DB)

機械語	ニーモニック	機械語	ニーモニック
B 0	OR B	C 6	ADD A, n
B 1	OR C	C 7	RST 0H
B 2	OR D	C 8	RET Z
B 3	OR E	C 9	RET
B 4	OR H	CA	JP Z, nn
B 5	OR L	CB	121~126 ページ参照
B 6	OR (HL)	CC	CALL Z, nn
B 7	OR A	CD	CALL nn
B 8	CP B	CE	ADC A, n
B 9	CP C	CF	RST 8H
BA	CP D	D 0	RET NC
BB	CP E	D 1	POP DE
BC	CP H	D 2	JP NC, nn
BD	CP L	D 3	OUT (n), A
BE	CP (HL)	D 4	CALL NC, nn
BF	CP A	D 5	PUSH DE
C 0	RET NZ	D 6	SUB n
C 1	POP BC	D 7	RST 10H
C 2	JP NZ, nn	D 8	RET C
C 3	JP nn	D 9	<u>EXX</u>
C 4	CALL NZ, nn	DA	JP C, nn
C 5	PUSH BC	DB	IN A, (n)

機械語	ニーモニック	機械語	ニーモニック
DC	CALL C, nn	F2	JP P, nn
DD	127ページ参照	F3	DI
DE	SBC A, n	F4	CALL P, nn
DF	RST 18H	F5	PUSH AF
E0	RET PO	F6	OR n
E1	POP HL	F7	RST 30H
E2	JP PO, nn	F8	RET M
E3	EX (SP), HL	F9	LD SP, HL
E4	CALL PO, nn	FA	JP M, nn
E5	PUSH HL	FB	EI
E6	AND n	FC	CALL M, nn
E7	RST 20H	FD	127ページ欄外参照
E8	RET PE	FE	CP n
E9	JP (HL)	FF	RST 38H
EA	JP PE, nn		
EB	EX DE, HL		
EC	CALL PE, nn		
ED	128~129ページ参照		
EE	XOR n		
EF	RST 28H		
F0	RET P		
F1	POP AF		

機械語→ニーモニック

(CB00~CB2B)

機械語	ニーモニック	機械語	ニーモニック
CB00	RLC B	CB16	RL (HL)
CB01	RLC C	CB17	RL A
CB02	RLC D	CB18	RR B
CB03	RLC E	CB19	RR C
CB04	RLC H	CB1A	RR D
CB05	RLC L	CB1B	RR E
CB06	RLC (HL)	CB1C	RR H
CB07	RLC A	CB1D	RR L
CB08	RRC B	CB1E	RR (HL)
CB09	RRC C	CB1F	RR A
CB0A	RRC D	CB20	SLA B
CB0B	RRC E	CB21	SLA C
CB0C	RRC H	CB22	SLA D
CB0D	RRC L	CB23	SLA E
CB0E	RRC (HL)	CB24	SLA H
CB0F	RRC A	CB25	SLA L
CB10	RL B	CB26	SLA (HL)
CB11	RL C	CB27	SLA A
CB12	RL D	CB28	SRA B
CB13	RL E	CB29	SRA C
CB14	RL H	CB2A	SRA D
CB15	RL L	CB2B	SRA E

機械語→ニーモニック

(CB2C~CB5F)

機械語	ニーモニック	機械語	ニーモニック
CB2C	SRA H	CB4A	BIT 1, D
CB2D	SRA L	CB4B	BIT 1, E
CB2E	SRA (HL)	CB4C	BIT 1, H
CB2F	SRA A	CB4D	BIT 1, L
CB38	SRL B	CB4E	BIT 1, (HL)
CB39	SRL C	CB4F	BIT 1, A
CB3A	SRL D	CB50	BIT 2, B
CB3B	SRL E	CB51	BIT 2, C
CB3C	SRL H	CB52	BIT 2, D
CB3D	SRL L	CB53	BIT 2, E
CB3E	SRL (HL)	CB54	BIT 2, H
CB3F	SRL A	CB55	BIT 2, L
CB40	BIT 0, B	CB56	BIT 2, (HL)
CB41	BIT 0, C	CB57	BIT 2, A
CB42	BIT 0, D	CB58	BIT 3, B
CB43	BIT 0, E	CB59	BIT 3, C
CB44	BIT 0, H	CB5A	BIT 3, D
CB45	BIT 0, L	CB5B	BIT 3, E
CB46	BIT 0, (HL)	CB5C	BIT 3, H
CB47	BIT 0, A	CB5D	BIT 3, L
CB48	BIT 1, B	CB5E	BIT 3, (HL)
CB49	BIT 1, C	CB5F	BIT 3, A

機械語→ニーモニック

(CB60~CB8B)

機械語	ニーモニック	機械語	ニーモニック
CB60	BIT 4, B	CB76	BIT 6, (HL)
CB61	BIT 4, C	CB77	BIT 6, A
CB62	BIT 4, D	CB78	BIT 7, B
CB63	BIT 4, E	CB79	BIT 7, C
CB64	BIT 4, H	CB7A	BIT 7, D
CB65	BIT 4, L	CB7B	BIT 7, E
CB66	BIT 4, (HL)	CB7C	BIT 7, H
CB67	BIT 4, A	CB7D	BIT 7, L
CB68	BIT 5, B	CB7E	BIT 7, (HL)
CB69	BIT 5, C	CB7F	BIT 7, A
CB6A	BIT 5, D	CB80	RES 0, B
CB6B	BIT 5, E	CB81	RES 0, C
CB6C	BIT 5, H	CB82	RES 0, D
CB6D	BIT 5, L	CB83	RES 0, E
CB6E	BIT 5, (HL)	CB84	RES 0, H
CB6F	BIT 5, A	CB85	RES 0, L
CB70	BIT 6, B	CB66	RES 0, (HL)
CB71	BIT 6, C	CB87	RES 0, A
CB72	BIT 6, D	CB88	RES 1, B
CB73	BIT 6, E	CB89	RES 1, C
CB74	BIT 6, H	CB8A	RES 1, D
CB75	BIT 6, L	CB8B	RES 1, E

機械語→ニーモニック

(CB8C~CBB7)

機械語	ニーモニック	機械語	ニーモニック
CB8C	RES 1, H	CBA2	RES 4, D
CB8D	RES 1, L	CBA3	RES 4, E
CB8E	RES 1, (HL)	CBA4	RES 4, H
CB8F	RES 1, A	CBA5	RES 4, L
CB90	RES 2, B	CBA6	RES 4, (HL)
CB91	RES 2, C	CBA7	RES 4, A
CB92	RES 2, D	CBA8	RES 5, B
CB93	RES 2, E	CBA9	RES 5, C
CB94	RES 2, H	CBAA	RES 5, D
CB95	RES 2, L	CBAB	RES 5, E
CB96	RES 2, (HL)	CBAC	RES 5, H
CB97	RES 2, A	CBAD	RES 5, L
CB98	RES 3, B	CBAE	RES 5, (HL)
CB99	RES 3, C	CBAF	RES 5, A
CB9A	RES 3, D	CBB0	RES 6, B
CB9B	RES 3, E	CBB1	RES 6, C
CB9C	RES 3, H	CBB2	RES 6, D
CB9D	RES 3, L	CBB3	RES 6, E
CB9E	RES 3, (HL)	CBB4	RES 6, H
CB9F	RES 3, A	CBB5	RES 6, L
CBA0	RES 4, B	CBB6	RES 6, (HL)
CBA1	RES 4, C	CBB7	RES 6, A

機械語→ニーモニック

(CBB8~CBE3)

機械語	ニーモニック	機械語	ニーモニック
CBB8	RES 7, B	CBCE	SET 1, (HL)
CBB9	RES 7, C	CBCF	SET 1, A
CBBA	RES 7, D	CBD0	SET 2, B
CBBB	RES 7, E	CBD1	SET 2, C
CBBC	RES 7, H	CBD2	SET 2, D
CBBD	RES 7, L	CBD3	SET 2, E
CBBE	RES 7, (HL)	CBD4	SET 2, H
CBBF	RES 7, A	CBD5	SET 2, L
CBC0	SET 0, B	CBD6	SET 2, (HL)
CBC1	SET 0, C	CBD7	SET 2, A
CBC2	SET 0, D	CBD8	SET 3, B
CBC3	SET 0, E	CBD9	SET 3, C
CBC4	SET 0, H	CBDA	SET 3, D
CBC5	SET 0, L	CBDB	SET 3, E
CBC6	SET 0, (HL)	CBDC	SET 3, H
CBC7	SET 0, A	CBDD	SET 3, L
CBC8	SET 1, B	CBDE	SET 3, (HL)
CBC9	SET 1, C	CBDF	SET 3, A
CBCA	SET 1, D	CBE0	SET 4, B
CBCB	SET 1, E	CBE1	SET 4, C
CBCC	SET 1, H	CBE2	SET 4, D
CB CD	SET 1, L	CBE3	SET 4, E

機械語→ニーモニック

(CBE4~CBFF)

機械語	ニーモニック	機械語	ニーモニック
CBE4	SET 4, H	CBFA	SET 7, D
CBE5	SET 4, L	CBFB	SET 7, E
CBE6	SET 4, (HL)	CBFC	SET 7, H
CBE7	SET 4, A	CBFD	SET 7, L
CBE8	SET 5, B	CBFE	SET 7, (HL)
CBE9	SET 5, C	CBFF	SET 7, A
CBEA	SET 5, D		
CBEB	SET 5, E		
CBEC	SET 5, H		
CBED	SET 5, L		
CBEE	SET 5, (HL)		
CBEF	SET 5, A		
CBF0	SET 6, B		
CBF1	SET 6, C		
CBF2	SET 6, D		
CBF3	SET 6, E		
CBF4	SET 6, H		
CBF5	SET 6, L		
CBF6	SET 6, (HL)		
CBF7	SET 6, A		
CBF8	SET 7, B		
CBF9	SET 7, C		

DDXXをFDXXにすると
iXがiYになる。

機械語→ニーモニック

インデックスレジスター関係
(DD09~DDF9)

機械語	ニーモニック	機械語	ニーモニック
DD09	ADD iX, BC	DD74	LD (iX+n), H
DD19	ADD iX, DE	DD75	LD (iX+n), L
DD21	LD iX, nn	DD77	LD (iX+n), A
DD22	LD (nn), iX	DD7E	LD A, (iX+n)
DD23	INC iX	DD86	ADD A, (iX+n)
DD29	ADD iX, iX	DD8E	ADC A, (iX+n)
DD2A	LD iX, (nn)	DD96	SUB (iX+n)
DD2B	DEC iX	DD9E	SBC A, (iX+n)
DD34	INC (iX+n)	DDA6	AND (iX+n)
DD35	DEC (iX+n)	DDAE	XOR (iX+n)
DD36	LD (iX+n), n	DDB6	OR (iX+n)
DD39	ADD iX, SP	DDBE	CP (iX+n)
DD46	LD B, (iX+n)	DDCB	130ページ参照
DD4E	LD C, (iX+n)	DDE1	POP iX
DD56	LD D, (iX+n)	DDE3	EX (SP), iX
DD5E	LD E, (iX+n)	DDE5	PUSH iX
DD66	LD H, (iX+n)	DDE9	JP (iX)
DD6E	LD L, (iX+n)	DDF9	LD SP, iX
DD70	LD (iX+n), B		
DD71	LD (iX+n), C		
DD72	LD (iX+n), D		
DD73	LD (iX+n), E		

機械語	ニーモニック	機械語	ニーモニック
ED40	IN B, (C)	ED5A	ADC HL, DE
ED41	OUT (C), B	ED5B	LD DE, (nn)
ED42	SBC HL, BC	ED5E	IM 2
ED43	LD (nn), BC	ED5F	LD A, R
ED44	NEG	ED60	IN H, (C)
ED45	RETN	ED61	OUT (C), H
ED46	IM 0	ED62	SBC HL, HL
ED47	LD I, A	ED63	LD (nn), HL
ED48	IN C, (C)	ED67	RRD
ED49	OUT (C), C	ED68	IN L, (C)
ED4A	ADC HL, BC	ED69	OUT (C), L
ED4B	LD BC, (nn)	ED6A	ADC HL, HL
ED4D	RETI	ED6B	LD HL, (nn)
ED4F	LD R, A	ED6F	RLD
ED50	IN D, (C)	ED72	SBC HL, SP
ED51	OUT (C), D	ED73	LD (nn), SP
ED52	SBC HL, DE	ED78	IN A, (C)
ED53	LD (nn), DE	ED79	OUT (C), A
ED56	IM 1	ED7A	ADC HL, SP
ED57	LD A, I	ED7B	LD SP, (nn)
ED58	IN E, (C)	EDA0	LDI
ED59	OUT (C), E	EDA1	CPI

機械語→ニーモニック

(EDA2~EDBB)

機械語	ニーモニック	機械語	ニーモニック
EDA2	INI		
EDA3	OUTI		
EDA8	LDD		
EDA9	CPD		
EDAA	IND		
EDAB	OUTD		
EDB0	LDIR		
EDB1	CPIR		
EDB2	INIR		
EDB3	OTIR		
EDB8	LDDR		
EDB9	CPDR		
EDBA	INDR		
EDBB	OTDR		

DDCBXXをFDCBXXとするとiXがiYになる。XX = n

機械語	ニーモニック	機械語	ニーモニック
DDCBXX06	R L C (i X + n)	DDCBXXB6	R E S 6, (i X + n)
DDCBXX0E	R R C (i X + n)	DDCBXXBE	R E S 7, (i X + n)
DDCBXX16	R L (i X + n)	DDCBXXC6	S E T 0, (i X + n)
DDCBXX1E	R R (i X + n)	DDCBXXCE	S E T 1, (i X + n)
DDCBXX26	S L A (i X + n)	DDCBXXD6	S E T 2, (i X + n)
DDCBXX2E	S R A (i X + n)	DDCBXXDE	S E T 3, (i X + n)
DDCBXX3E	S R L (i X + n)	DDCBXXE6	S E T 4, (i X + n)
DDCBXX46	B I T 0, (i X + n)	DDCBXXEE	S E T 5, (i X + n)
DDCBXX4E	B I T 1, (i X + n)	DDCBXXF6	S E T 6, (i X + n)
DDCBXX56	B I T 2, (i X + n)	DDCBXXFE	S E T 7, (i X + n)
DDCBXX5E	B I T 3, (i X + n)		
DDCBXX66	B I T 4, (i X + n)		
DDCBXX6E	B I T 5, (i X + n)		
DDCBXX76	B I T 6, (i X + n)		
DDCBXX7E	B I T 7, (i X + n)		
DDCBXX86	R E S 0, (i X + n)		
DDCBXX8E	R E S 1, (i X + n)		
DDCBXX96	R E S 2, (i X + n)		
DDCBXX9E	R E S 3, (i X + n)		
DDCBXXA6	R E S 4, (i X + n)		
DDCBXXAE	R E S 5, (i X + n)		

		A	B	C	D	E	H	L
8 ビ ット 転 送 ・ 演 算	LD A	7 F	7 8	7 9	7 A	7 B	7 C	7 D
	LD B	4 7	4 0	4 1	4 2	4 3	4 4	4 5
	LD C	4 F	4 8	4 9	4 A	4 B	4 C	4 D
	LD D	5 7	5 0	5 1	5 2	5 3	5 4	5 5
	LD E	5 F	5 8	5 9	5 A	5 B	5 C	5 D
	LD H	6 7	6 0	6 1	6 2	6 3	6 4	6 5
	LD L	6 F	6 8	6 9	6 A	6 B	6 C	6 D
	LD (HL)	7 7	7 0	7 1	7 2	7 3	7 4	7 5
	LD (BC)	0 2						
	LD (DE)	1 2						
	LD (nn)	3 2						
	LD I	ED47						
	LD R	ED4F						
	ADD	8 7	8 0	8 1	8 2	8 3	8 4	8 5
	ADC	8 F	8 8	8 9	8 A	8 B	8 C	8 D
	SUB	9 7	9 0	9 1	9 2	9 3	9 4	9 5
	SBC	9 F	9 8	9 9	9 A	9 B	9 C	9 D
	AND	A 7	A 0	A 1	A 2	A 3	A 4	A 5
	XOR	A F	A 8	A 9	A A	A B	A C	A D
	OR	B 7	B 0	B 1	B 2	B 3	B 4	B 5
	CP	B F	B 8	B 9	B A	B B	B C	B D
	INC	3 C	0 4	0 C	1 4	1 C	2 4	2 C
	DEC	3 D	0 5	0 D	1 5	1 D	2 5	2 D

*インデックスレジスター関係は127ページ, 130ページを見よ

ニーモニック
↓
機械語

		(HL)	n	(BC)	(DE)	(nn)	I	R
8 ビット 転送・演算	LD A	7 E	3 E	0 A	1 A	3 A	ED57	ED5F
	LD B	4 6	0 6					
	LD C	4 E	0 E					
	LD D	5 6	1 6					
	LD E	5 E	1 E					
	LD H	6 6	2 6					
	LD L	6 E	2 E					
	LD (HL)		3 6					
	LD (BC)							
	LD (DE)							
	LD (nn)							
	LD I							
	LD R							
	ADD	8 6	C 6					
	ADC	8 E	C E					
	SUB	9 6	D 6					
	SBC	9 E	D E					
	AND	A 6	E 6					
	XOR	A E	E E					
	OR	B 6	F 6					
	CP	B E	F E					
	INC	3 4						
	DEC	3 5						

*インデックスレジスター関係は127ページ, 130ページを見よ

ニ
ー
モ
ニ
ツ
ク
↓
機
械
語

		AF	BC	DE	HL	SP	nn	(nn)
16 ビット 転送・演算	PUSH	F5	C5	D5	E5			
	POP	F1	C1	D1	E1			
	LD BC						01	ED4B
	LD DE						11	ED5B
	LD HL						21	2A
	LD SP				F9		31	ED7B
	LD (nn)		ED43	ED53	22	ED73		
	ADD HL		09	19	29	39		
	ADC HL		ED4A	ED5A	ED6A	ED7A		
	SBC HL		ED42	ED52	ED62	ED72		
	INC		03	13	23	33		
	DEC		0B	1B	2B	3B		

注 LD HL, (nn) は2A。しかし, ED6Bでも使える。また, LD (nn), HLの22はED63でも使える。
インテルの8080 (Z80のもとになったCPU) の命令は2A, 22なので, こちらを使った方が無難かもしれない。

入 出 力	IN A, (n)	DB
	OUT (n), A	D3
	IN A, (C)	ED78
	IN B, (C)	ED40
	IN C, (C)	ED48
	IN D, (C)	ED50
	IN E, (C)	ED58
	IN H, (C)	ED60
	IN L, (C)	ED68
	OUT (C), A	ED79
	OUT (C), B	ED41
	OUT (C), C	ED49
	OUT (C), D	ED51
	OUT (C), E	ED59
	OUT (C), H	ED61
	OUT (C), L	ED69

ブ ロ ッ ク 入 出 力 ・ 転 送 ・ 検 索	INI	EDA2
	INIR	EDB2
	IND	EDAA
	INDR	EDBA
	OUTI	EDA3
	OTIR	EDB3
	OUTD	EDAB
	OTDR	EDBB
	LDI	EDA0
	LDIR	EDB0
	LDD	EDA8
	LDDR	EDB8
	CPI	EDA1
	CPIR	EDB1
	CPD	EDA9
	CPDR	EDB9

ジ ャ ン プ	JR	n	18
	JR	Z, n	28
	JR	NZ, n	20
	JR	C, n	38
	JR	NC, n	30
	JP	nn	C3
	JP	Z, nn	CA
	JP	NZ, nn	C2
	JP	C, nn	DA
	JP	NC, nn	D2
	JP	PE, nn	EA
	JP	P0, nn	E2
	JP	P, nn	F2
	JP	M, nn	FA
コ ー ル	CALL	nn	CD
	CALL	Z, nn	CC
	CALL	NZ, nn	C4
	CALL	C, nn	DC
	CALL	NC, nn	D4
	CALL	PE, nn	EC
	CALL	P0, nn	E4
	CALL	P, nn	F4
	CALL	M, nn	FC

リ タ ー ン	RET	C9
	RET Z	C8
	RET NZ	C0
	RET C	D8
	RET NC	D0
	RET PE	E8
	RET PO	E0
	RET P	F0
	RET M	F8
	RETI	ED4D
	RETN	ED45
リ ス タ ー ト	RST 0H	C7
	RST 8H	CF
	RST 10H	D7
	RST 18H	DF
	RST 20H	E7
	RST 28H	EF
交 換	RST 30H	F7
	RST 38H	FF
	EX DE, HL	EB
	EX AF, AF'	08
	EXX	D9
交 換	EX (SP), HL	E3

操 作 ・ 特 殊 J P	DAA	2 7
	CPL	2 F
	NEG	ED44
	CCF	3 F
	SCF	3 7
	JP (HL)	E 9
	DJNZ n	1 0
C P U コ ン ト ロ ー ル	EI	FB
	DI	F 3
	IM 0	ED46
	IM 1	ED56
	IM 2	ED5E
	NOP	0 0
	HALT	7 6
回 転	RLD	ED6F
	RRD	ED67
	RLCA	0 7
	RRCA	0 F
	RLA	1 7
	RRA	1 F

週1回2時間 合計15回でのカリキュラム例

1	スイッチの入れかた キーボードの説明 PCテスト法 (FRE BEEP TIME\$)ダイレクトモード (PRINT 四則演算 コンマとセミコロン) プログラムモード (INPUT LET END) RETURNとNEWの徹底 電源を一度切ったら5秒以上たたなければ再度スイッチを入れてはいけないこととRESETを押すと初期状態にもどることの徹底 キャラクターモードのLINEでの線引き
2	復習 ファンクションキーの再徹底 プログラムの修正方法 円の面積 (GOTO IF~THEN STOP) フローチャートの説明 キャラクターモードのLINEで箱を描く 次回なにをやるか予告すること
3	復習 5個のデータの平均など平均を求めるプログラム FOR~NEXTによる繰り返し処理 PSETを使った線引き X軸Y軸を引きPSETで任意の位置に点を打つ 初心者の第1難関はFOR~NEXT
4	復習 FOR~NEXTはフローチャートを描いて再徹底 九九の掛け算で多重ループの説明 READ DATA TABによる二次方程式 FOR~NEXTとPSET PRESETを組み合わせて線 ドットが進む
5	READ DATAの復習 GOSUBとRETURN REMとアポストロフィでの注釈行の入れ方 RESTORE 図形 (STRING\$を使ってみる)
6	復習 変数の説明 (単精度 倍精度 整数 文字変数などをくわしく) 配列 (DIM) 図形はKの長さの線が進む 出席者が減りだす 抜けた人はついてくることができなくなる そのひとつにはキーボード操作の習熟を
7	復習 二重添字つき変数 図形は一步後退二歩前進など複雑な動きをするもの 配列に時間をかけること このあたりからキーボード操作が速くなり いい質問が出はじめる さもないと講義は形式的に進行していることになる
8	復習 作表 (PRINT USING) RNDとINT PRINT USINGでレイアウトすることを十分理解させる
9	復習 情報検索 (INSTR をていねいに説明する 番号から名前を求める その逆 文字検索の意味を理解させる) 出席者はまったく固定する 図形はランダムに箱を描く、箱の中で玉が動くなどを適当に
10	復習 分類 (データがどう比較されて入れかわるか経過を出力すると理解が早い 黒板に書いてもよい 文字ソートもできること SWAPも試してみる)
11	復習 関数 (画面との対話が主だったのでフローチャートからプログラムを書かせてみると30分はかかる) 図形は円 サインカーブ
12	復習 関数の自由定義 (借金返済 元利合計など具体的なものを) 乱数の作りかたの例 図形をやる時間はないだろう

13	復習 2進10進16進 キャラクターコード表の説明 文字から数値へ 数値から文字への変換 プログラムをとりおく (CLOAD CSAVE など デモ用カ セットを用意する)
14	復習 折れ線グラフなど (受講者のテーマに合わせる)
15	復習 ヒストグラムなど (受講者のテーマに合わせる)

*講習の注意点

- (1) 2時間を1時間30分と30分に分ける。後半の30分を図形にあてる。
- (2) 1つのポイントは『キーボードをいかに速く習熟させるか』にある。カナ文字を使ったプログラムは講義を遅らせる。あらかじめ、キーボード図を渡し
ておく。大きなキーボード図も掲示用につくっておく。
- (3) ファンクションキーを最初に教えておくと入力がかなり楽になる。
特に、RETURNキーは、必ず押すような癖をつけさせる。
- (4) 常識外のエラーはすべてキーボードの誤操作にある。
- (5) 予習復習をさせないと、週1回の講義では忘れてしまう。次回なにをやるか
予告すること。
- (6) プリンターがあれば、あらかじめ当日のプログラム・リストを受講生分だけ
作ってやるくらいの親切さは欲しい。ノートをとることに熱中する人がいる。
- (7) プログラムはあらかじめカセットテープにとっておいて、それを使う手もあ
る。
- (8) 15回では、機械語に連結していくUSR関数までやることはできない。
- (9) 他クラスのひとが振り替え受講するような場合、振り替え受講者には機械を
さわらせない。複数クラスが同時進行している場合には、講師間で意思統一
して、1週間通して同じ講義内容にする。月曜第1回の講師は必ずベテラン
をあてて、できれば他の講師も受講する態度が望ましい。
- (10) 機械台数が少ない場合には、講師がこまめに交替を指示すること。
- (11) 受講者のレベルによっては、第7回あたりから図形の時間が足りなくなる。
そのときには、第14回、第15回の余裕をにらんで、適宜スケジュールの
調整をする。
- (12) OAのための企業内講座の場合には、このBASIC講座が逆効果になって
『日暮れて道遠し』と思わせてはいけない。いわゆる汎用事務計算パッケ
ジプログラムを用意しておいて、高級電卓程度の仕事は簡単にできることを
認識させる。
- (13) 『知っていること』と『教えること』とは違う。やさしく説明できる技術は
事前の十分な練習によってしか身につかない。しゃべり過ぎに注意する。
- (14) 受講者は、講師の想像以上に神経を使っている。早め早めに小休止をいれて
背伸び、首の運動、指の運動などをする。
- (15) 受講者の記録は必ずとる。どこで行き詰まったかをみると、講師の力不足が
原因だと気づく。謙虚に反省して次回に備えよう。

① PCのテスト法

```
PRINT FRE (0) .....16Kの場合10402とでる。  
                      32Kの場合26786とでる。
```

```
BEEP 1 .....ブザーが鳴る。
BEEP 0 .....とまる。
```

PRINT TIME\$.....スイッチを入れてからの時間がでる。

② ダイレクトモード

PRINT "ASAHI"

```
PRINT 2 + 3, 6 - 3, 5 * 3, 6 / 3
```

PRINTの省略記号は疑問符 (?)

PRINT文中のコンマ (,) とセミコロン (;) のちがいを

③ プログラムモード

```
10 INPUT A
20 PRINT A
30 END
```

```
10 INPUT A
20 LET B=A
30 PRINT B
40 END
```

```

10 INPUT A, B
20 LET C=A*B
30 PRINT "A*B=" ; C
40 END

```

④ 図形 線を描く

LINE (20, 0) - (20, 15), "A", 0

LINE (0, 12) - (35, 12), "◆", 0

LINE (5, 7) - (30, 20), " * ", 0

① ↑ ↓ → ← INSキー DELキーの使いかた

② 円の面積

```
10 INPUT R
20 M=3.14159*R*R
30 PRINT M
40 END
```

```
10 INPUT R
20 M=3.14159*R^2
30 PRINT M
40 GOTO 10
50 END
```

```
10 INPUT R
20 IF R=0 THEN GOTO 60
30 M=3.14159*R*R
40 PRINT M
50 GOTO 10
60 END
```

③ STOP

```
10 INPUT R
20 A=R*R
30 STOP
40 M=3.14159*A
50 PRINT M
60 END
```

上のプログラムは30行でとまる。ダイレクトモードでPRINT A を実行するとR*Rの答が出る。CONTと入れてRETURNキーを押すとプログラムの実行が40行から再開される。STOPはデバック (Debug) などを使う。

④ 図形 箱を描く

```
10 LINE (5, 5) - (35, 20), " * ", B
10 LINE (5, 5) - (35, 20), " * ", BF

10 WIDTH 80, 25
20 LINE (30, 10) - (100, 90), PSET, 0
30 LINE (35, 15) - (80, 75), PSET, B
40 LINE (40, 20) - (70, 65), PSET, BF
```


① 平均

```

10 A=1+2+3+4+5
20 PRINT A/5
30 END

10 INPUT A, B, C, D, E
20 PRINT (A+B+C+D+E) /5
30 GOTO 10

```

② ゼロを入れると平均を出力する

```

10 K=0 : S=0
20 INPUT "データ ヲ イレナサイ" ; X
30 IF X=0 THEN 70
40 K=K+1
50 S=S+X
60 GOTO 20
70 PRINT "データ スウ=" ; K
80 PRINT "ゴウケイ=" ; S
90 PRINT "ヘイキン=" ; S/K
100 PRINT
110 END

```

③ 繰り返し

<p>a.</p> <pre> 10 X=1 20 PRINT X; 30 X=X+1 40 IF X<=30 THEN 20 50 END </pre>	<p>b.</p> <pre> 10 FOR X=1 TO 30 20 PRINT X; 30 NEXT X 40 END </pre>
--	--

aとbの結果は、同じになる。bの10行をFOR X =1 TO 30 STEP 2 としてみよ。

④ 図形 X軸とY軸を引き、任意の位置に点(ドット)を出す

```

10 WIDTH 80, 25
20 LINE (80, 0) - (80, 79), PSET, 0
30 LINE (0, 40) - (157, 40), PSET, 0
40 INPUT X
50 INPUT Y
60 PSET (X, Y, 0)
70 GOTO 40

```

- ① 100から1まで3ずつ減らしていく

```
10 FOR J=100 TO 1 STEP -3
20 PRINT J;
30 NEXT J
40 END
```

- ② 九九の掛け算

```
10 FOR K=1 TO 9
20 FOR J=1 TO 9
30 PRINT J*K;
40 NEXT J
50 PRINT
60 NEXT K
70 END
```

- ③ READとDATA

```
10 READ X, Y, Z
20 A=X+Y+Z
30 PRINT X;Y;Z;A
40 DATA 7, 5, 3
50 END
```

READ文の変数の数とDATA文のデータの数と一致していることに注意。

- ④ 二次方程式

```
10 FOR X=-6 TO 6
20 PRINT TAB (X*X) ; "●"
30 NEXT X
40 END
```

- ⑤ 図形 点(ドット)が進む

```
10 WIDTH 80, 25
20 FOR X=-40 TO 40
30 Y=X
40 FOR J=1 TO 5
50 PSET (X+80, Y+40, 0)
60 NEXT J
70 PRESET (X+80, Y+40)
80 NEXT X
90 GOTO 20
```

① 2つの数を入れて足し算をしてみる

```

10 INPUT A, B
20 GOSUB 100
30 PRINT A+B
40 IF (A+B) < 100 THEN 80
50 FOR J=1 TO 800
60 BEEP1:BEEP0 ' 100 >= (A+B) ノ トキ
70 NEXT J
80 GOSUB 100
90 END
100 REM*****SUBROUTINE*****
110 BEEP1
120 FOR K=1 TO 800:NEXT K
130 BEEP0
140 RETURN

```

② RESTORE

```

10 READ A, B, C
20 READ D, E, F
30 PRINT A+B, D+E, C+F
40 DATA 2, 4, 6
50 DATA 2, 4, 6
60 END

```

```

10 READ A, B, C
20 RESTORE
30 READ D, E, F
40 PRINT A+B, D+E, C+F
50 DATA 2, 4, 6
60 END

```

RESTORE 90 などのように、行指定もできる。

③ 図形 ●で三角を描く

```

10 FOR J=1 TO 6
20 PRINT J ; TAB (4) ; "カイ" ; TAB (9) ; STRING$ (J, "●")
30 NEXT J
40 K=6
50 FOR J=6 TO 1 STEP -1
60 K=K+1
70 PRINT K ; TAB (4) ; "カイ" ; TAB (9) ; STRING$ (J, "●")
80 NEXT J

```


① 変数

```

10 A$="ASAHI"
20 B$="WEEKLY"
30 PRINT A$+B$

10 INPUT R#
20 M#=3.14159*R#^2
30 PRINT M#

```

② 配列 (添字つき変数)

```

10 DIM X(10)
20 FOR J=1 TO 10
30 X(J)=J*2
40 NEXT J
50 FOR K=1 TO 10
60 PRINT X(K);
70 NEXT K

```

DIM X(10) で X(0), X(1),X(10) まで11個の場所が確保される。上の例では, X(0) は使っていない。

```

10 INPUT N
20 DIM X(N)
30 FOR J=1 TO N
40 PRINT "X=";
50 INPUT X(J)
60 NEXT J
70 PRINT
80 T=0
90 FOR J=1 TO N
100 T=T+X(J)
110 NEXT J
120 PRINT "GOKEI=";T

```

③ 図形 Kの長さの線が進む

```

10 WIDTH 80, 25
20 FOR X=-40 TO 40:Y=X
30 FOR K=0 TO 7:PSET(X+80+K, Y+40+K, 0)
40 NEXT K
50 PRESET(X+80, Y+40, 0):NEXT X
60 FOR K=0 TO 7
70 PRESET(120+K, 80+K, 0)
80 NEXT K

```

① 2重添字つき変数

```

10 DIM A (2, 3)
20 FOR X=0 TO 2
30 FOR Y=0 TO 3
40 READ A (X, Y)
50 NEXT Y
60 NEXT X
70 FOR X=0 TO 2
80 FOR Y=0 TO 3
90 PRINT A (X, Y)
100 NEXT Y
110 NEXT X
120 DATA 1, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22

```

60行までが実行されると、下のように読み込まれる。

A (0, 0) 1	A (0, 1) 2	A (0, 2) 4	A (0, 3) 6
A (1, 0) 8	A (1, 1) 10	A (1, 2) 12	A (1, 3) 14
A (2, 0) 16	A (2, 1) 18	A (2, 2) 20	A (2, 3) 22

70行と80行, 100行と110行を入れ換えてみよ。

② 図形 1歩後退2歩前進

```

10 WIDTH 80, 25
20 FOR X=-40 TO 40
30 Y=X
40 FOR K=0 TO 4
50 PSET (X+80+K, Y+40+K, 0)
60 FOR J=1 TO 10:NEXT J
70 PRESET (X+80+K, Y+40+K, 0)
80 NEXT K
90 NEXT X

```

① 作表

	サトウ	タナカ	キムラ	モリタ
エイゴ	56	94	47	82
コクゴ	72	63	93	79
メンセツ	83	58	71	67

```

10 DIM N$ (4), K$ (2), A (2, 3)
20 DATA " ", " ", "サトウ", "タナカ", "キムラ"
30 DATA "モリタ"
40 DATA "エイゴ", "コクゴ", "メンセツ"
50 DATA 56, 94, 47, 82, 72, 63, 93, 79, 83
60 DATA 58, 71, 67
70 FOR J=0 TO 4:READ N$ (J):NEXT J
80 FOR J=0 TO 2:READ K$ (J):NEXT J
90 FOR X=0 TO 2:FOR Y=0 TO 3
100 READ A (X, Y):NEXT Y:NEXT X
110 FOR J=0 TO 4
120 PRINT USING "& &";N$ (J);:NEXT J
130 PRINT
140 FOR X=0 TO 2
150 PRINT USING "& &";K$ (X);
160 FOR Y=0 TO 3
170 PRINT USING "####";A (X, Y);:NEXT Y
180 PRINT
190 NEXT X

```

② RND (乱数) と INT (整数にする) で◆やサイコロの目を出す

```

10 N=INT (15*RND (1) +1)
20 PRINT STRING$ (N, "◆")
30 GOTO 10

10 FOR J=1 TO 10
20 PRINT INT (6*RND (1) +1);
30 NEXT J

```


① 検索 コードで名前を探す

```

10 DIM S (20), S$ (20)
20 GOSUB 200
30 INPUT "コード ヲ ドウゾ" ; S
40 FOR J=1 TO 20
50 IF S (J) = S THEN 100
60 NEXT J
70 PRINT
80 PRINT "コード ナシ"
90 GOTO 30
100 PRINT "コード" ; S ; " ナマエ " ; S$ (J)
110 PRINT
120 GOTO 30
130 END
200 FOR J=1 TO 10
210 READ S (J), S$ (J)
220 NEXT J
230 RETURN
300 DATA 100, サトウ, 200, タナカ, 300, キムラ, 400
310 DATA コイケ, 500, アラキ, 600, ウスイ, 700, トミタ
320 DATA 800, ムライ, 900, カネコ, 910, ナカノ

```

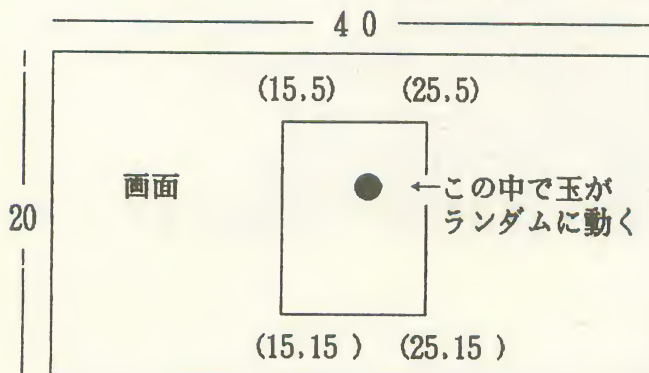
② 名前でコードを探すには、次のように変更する

```

30 INPUT "ナマエ ヲ ドウゾ" ; S$
50 IF INSTR (S$ (J), S) <> 0 THEN 100
100 PRINT "ナマエ " ; S$ (J) ; " コード" ; S (J)

```

③ 図形 箱の中で玉が動く プログラムは次のページにある



③のつづき

```

10 REM ***** ハコ ノ ナカ デ タマ ガ ウゴク *****
20 LINE (15, 6) - (15, 14), " | "
30 LINE (25, 6) - (25, 14), " | "
40 LINE (16, 5) - (24, 5), " — "
50 LINE (16, 15) - (24, 15), " — "
60 LOCATE 15, 5:PRINT "┌"
70 LOCATE 15, 15:PRINT "└"
80 LOCATE 25, 5:PRINT "┐"
90 LOCATE 25, 15:PRINT "┘"
100 X=20
110 Y=10
120 LOCATE X, Y:PRINT "●"
130 BEEP 1:FOR J=1 TO 50:BEEP 0:NEXT J
140 LOCATE X, Y:PRINT " "
150 FOR J=1 TO 20:NEXT J
160 X=X+INT (RND (3) * 3) - 1
170 Y=Y+INT (RND (3) * 3) - 1
180 REM *****ハコ カラ デナイヨウニスル*****
190 IF X<16 THEN X=X+1
200 IF X>24 THEN X=X-1
210 IF Y<6 THEN Y=Y+1
220 IF Y>14 THEN Y=Y-1
230 GOTO 120

```

次の行を追加すると、XとYの位置（玉の位置）がわかる。

```

185 LOCATE 1, 1:PRINT X, Y

```

④ 図形 ランダムに箱を描く（関数の定義は第12回でやる）

```

10 WIDTH 80, 25:CONSOLE 0, 25, 0, 1
20 DEFFNR (R) = RND (1) * R
30 COLOR, 0, 1:PRINT CHR$ (12)
40 FOR J=1 TO 50
50 COLOR (INT (RND (3) * 7 + 1))
60 X1=FNR (159):X2=FNR (159)
70 Y1=FNR (100):Y2=FNR (100)
80 LINE (X1, Y1) - (X2, Y2), PSET, B
90 NEXT J
100 END

```

① 分類 (ならべかえ)

```

10 DIM S (10), S$ (10)
20 GOSUB 200
30 FOR J=1 TO 9
40 FOR K=1 TO 10-J
50 IF S (K) <= S (K+1) THEN 80
60 SWAP S (K), S (K+1)
70 SWAP S$ (K), S$ (K+1)
80 NEXT K
90 GOSUB 300
100 NEXT J
110 END
200 FOR L=1 TO 10
210 READ S (L), S$ (L)
220 NEXT L
230 RETURN
240 DATA 3, C, 6, F, 10, J, 4, D, 2, B, 7, G
250 DATA 5, E, 1, A, 9, I, 8, H
300 FOR M=1 TO 10
310 PRINT S (M) ; S$ (M) ;
320 NEXT M
330 PRINT
340 RETURN

```

SWAPを使わなければ、次のようになる。

```

60 T=S (K) : T$=S$ (K)
65 S (K)=S (K+1) : S$ (K)=S$ (K+1)
70 S (K+1)=T : S$ (K+1)=T$

```

② 最小値を求める

```

10 INPUT "カズ ノ コスウ" ; N
20 DIM X (N)
30 FOR J=1 TO N
40 INPUT "X=" ; X (J)
50 NEXT J
60 PRINT
70 M=X (1)
80 FOR K=1 TO N
90 IF M <= X (K) THEN 110
100 M=X (K)
110 NEXT K
120 PRINT "サイショウチ=" ; M
130 END

```


① 関数

```
10 FOR J=1 TO 40
20 PRINT SQR (J), RND (J), EXP (J)
30 NEXT J
40 END
```

② 円を描く

```
10 WIDTH 80, 25
20 P=3.14159
30 FOR T=0 TO 2*P STEP .03
40 X=30*COS (T) +40
50 Y=30*SIN (T) +50
60 LINE (X, Y) - (40, 50), PSET
70 X1=X+80:Y1=Y
80 PSET (X1, Y1, 0)
90 NEXT T
100 END
```

40行を $X=40*\cos(T)+40$ にしてみよ。

③ うずまきを描く

```
10 WIDTH 80, 25
20 FOR T=1 TO 1440
30 X=COS (T*3.14159/180)
40 Y=SIN (T*3.14159/180)
50 PSET (T/40*X+80, T/40*Y+40, 0)
60 NEXT T
70 END
```

④ テンキーの2, 4, 6, 8を使って線を描く (キーボードマップ参照のこと)

```
10 WIDTH 80, 25:COLOR, 0, 1
20 PRINT CHR$(12):X=80:Y=50
30 PSET (X, Y)
40 IF INP (0) =251 THEN Y=Y+1
50 IF INP (0) =239 THEN X=X-1
60 IF INP (0) =191 THEN X=X+1
70 IF INP (1) =254 THEN Y=Y-1
80 IF INP (1) =127 THEN END
90 GOTO 30
```

① 関数の自由定義

円の面積は第2回でやったとおり $M = 3.14159 * R * R$ だった。
MをRの関数という。ここで、関数関係を明確にするために次のようにする。

DEF FN M (R) = 3.14159 * R * R

使うときには、FN M (式) の形で使う。

```
10 DEF FN M (R) = 3.14159 * R * R
20 X = FN M (5)
30 A = 10
40 Y = FN M (A)
50 Z = FN M (A + 5)
60 PRINT X, Y, Z
70 END
```

② 元利合計を求める

```
10 DEF DBL N = S
20 DEF FNS (P, N, R) = P * (1 + R) ^ N
30 INPUT "ガンキン=" ; P
40 INPUT "リリツ=" ; R
50 INPUT "ネンスウ=" ; N
60 PRINT
70 PRINT "ガンリゴウケイ=" ; FNS (P, N, R)
80 END
```

③ 乱数を発生させて条件で行き先をかえる

```
10 WIDTH 80, 25 : PRINT CHR$ (12)
20 IF A = 75 OR B = 75 OR C = 75 OR D = 75 THEN 500
30 R = INT (4 * RND (1) + 1)
40 ON R GOTO 100, 200, 300, 400
100 A = A + 1 : LOCATE 0, 0 : PRINT STRING$ (A, "●") : GOTO 20
200 B = B + 1 : LOCATE 0, 4 : PRINT STRING$ (B, "○") : GOTO 20
300 C = C + 1 : LOCATE 0, 8 : PRINT STRING$ (C, "◆") : GOTO 20
400 D = D + 1 : LOCATE 0, 12 : PRINT STRING$ (D, "■") : GOTO 20
500 LOCATE 0, 16 : PRINT MID$ ("●○◆■", R, 1) ; "ノカチ"
```

④ 乱数の作りかたの例

```
1と-1をつくる INT (RND (3) * 2) * 2 - 1
1と0と-1を出す INT (RND (1) * 3) - 1
25から54を出す INT (RND (1) * 30) + 25
```

① 2進・10進・16進

10進数 2進数 16進数

00000.....	0
10001.....	1
20010.....	2
30011.....	3
40100.....	4
50101.....	5
60110.....	6
70111.....	7
81000.....	8
91001.....	9
101010.....	A
111011.....	B
121100.....	C
131101.....	D
141110.....	E
151111.....	F

8の位 4の位 2の位 1の位
 ↑ ↑ ↑ ↑
 (Arrows from 10, 11, 12, 13, 14, 15 to 8, 4, 2, 1 bits)

(1) 2進数の1桁をビットという。

(2) 2進数のビットをぜんぶ足すと10進数になる。

$$0111 = 0 + 4 + 2 + 1 = 7$$

(3) 2進数8桁の最大は

$$11111111 = 255 = FF$$

(Diagram showing two groups of four 1s, each labeled F)

② 10進数を2進数に変換する

```

10  DIM A (16) : B=0 : C=0 : J=0
20  INPUT "10シンスウ=" ; D
30  T=D
40  B=T/2 : C=T MOD 2
50  A(J)=C
60  IF B=1 THEN A(J+1)=1 : GOTO 90
70  T=B
80  IF B=0 THEN GOTO 90 ELSE J=J+1 : GOTO 40
90  PRINT D ; "=" ;
100 FOR K=J+1 TO 0 STEP -1
110 PRINT USING "#"; A(K) ;
120 NEXT K
130 PRINT : ERASE A : GOTO 10
  
```

③ 1文字をコードにコードを文字に変換する (キャラクターコード表参照)

```

10  PRINT ASC ("B")
20  PRINT CHR$(65)
30  FOR J=45 TO 255 : PRINT CHR$(J) ; : NEXT J
40  END
  
```


① 折れ線グラフ

```

1 0  WIDTH 80.25 : COLOR 0,0,1
2 0  DIM AX (145) , AY (90)
3 0  INPUT "データ / コスウ" ; KX
4 0  INPUT "データ / サイショウチ" ; YS
5 0  INPUT "データ / サイダイチ" ; YE
6 0  INPUT "メモリ / ハバ" ; KY
7 0  JX = (146 / KX)
8 0  IF JX < 2 THEN 30
9 0  IY = (YE - YS) / KY
1 0 0  JY = 90 / (YE - YS)
1 1 0  IF 90 / KY < 2 THEN 40
1 2 0  PRINT CHR$ (12)
1 3 0  LINE (10,90) — (159,90) ,PSET
1 4 0  LINE (10,90) — (10,0) ,PSET
1 5 0  PX = 10
1 6 0  FOR J = 1 TO KX
1 7 0  PX = PX + JX
1 8 0  PSET (PX,91)
1 9 0  NEXT J
2 0 0  FOR J = 1 TO IY
2 1 0  PY = PY + JY * KY
2 2 0  PSET (9,90 - PY)
2 3 0  LOCATE 0, (90 - PY) / 4
2 4 0  PRINT USING "###" ; YS + KY * J
2 5 0  NEXT J
2 6 0  PX = 10 + JX
2 7 0  FOR J = 1 TO KX
2 8 0  LOCATE 60,1
2 9 0  INPUT "データ" ; Y
3 0 0  YY = 90 - JY * (Y - YS)
3 1 0  PSET (PX,YY)
3 2 0  AX (J) = PX
3 3 0  AY (J) = YY
3 4 0  PX = PX + JX
3 5 0  NEXT J
3 6 0  FOR J = 2 TO KX
3 7 0  LINE (AX (J - 1) , AY (J - 1)) — (AX (J) , AY (J)) ,PSET
3 8 0  NEXT J
3 9 0  END

```

データの個数
縦軸の最小値
縦軸の最大値
縦軸の目盛りの幅
横軸の画面上のきざみ

縦軸の目盛りの個数
縦軸の画面上のきざみ

画面クリアー
横軸を描く
縦軸を描く

横軸に10からはじめて J X
の幅できざみ点をつける

縦軸に10からはじめて
J Y * K Y の幅で点と目盛
りをつける

データの入力

点をうつ
点の位置を保存する

次のきざみへいく

点を線でつなぐ

3 9 0 GOTO 260 とすると複数の線を描くことができる。

① ヒストグラム

```

1 0  WIDTH 80,25 :PRINT CHR$ (12)
2 0  DIM D (10)
3 0  LOCATE 5,0 :PRINT " **HISTGRAM** "
4 0  FOR I =2 TO 13
5 0      LOCATE 15,I :PRINT " | "
6 0      LOCATE 40,I :PRINT " . "
7 0      LOCATE 65,I :PRINT " . "
8 0  NEXT I
9 0  FOR I =15 TO 70
10 0      LOCATE I,13 :PRINT " — "
11 0  NEXT I
12 0  L =3
13 0  FOR I =1 TO 10
14 0      LOCATE 0,L :PRINT "rank" ; I
15 0      LOCATE 8,L :PRINT USING "##" ; D (I )
16 0      L =L+1
17 0  NEXT I
18 0  LOCATE 0,15 :PRINT "total "
19 0  LOCATE 7,15 :PRINT USING "##" ; D (0 )
20 0  FOR I =0 TO 50 STEP 10
21 0      LOCATE I+14 ,14 :PRINT USING "##" ; I
22 0  NEXT I
23 0  LOCATE 0,20 :PRINT "←スペース 1 1 個→"
24 0  LOCATE 0,20 :INPUT "DATA" ; X$
25 0  IF X$ ="end " THEN 370
26 0  X = VAL ( X$ )
27 0  IF X >=0 AND X <=100 THEN 290
28 0  LOCATE 0,21 :PRINT "DATA error" :GOTO 230
29 0  LOCATE 0,21 :PRINT "←スペース 1 7 個→"
30 0  IF X=100 THEN J=10 :GOTO 320
31 0  J = INT ( X / 10 ) + 1
32 0  D (J) = D (J) +1 : D (0) = D (0) +1
33 0  LOCATE 8,J+2 :PRINT USING "##" ; D (J)
34 0  LOCATE 15+ D (J) , J+2 :PRINT "■"
35 0  LOCATE 7,15 :PRINT USING "###" ; D (0 )
36 0  GOTO 230
37 0  END

```

縦の線

横の線

ランク

合計

横の刻み,0,10,20..50

データのインプットは
ENDで終わり

X>100,X <0 は
データエラー

インプットした値を
配列の各ランクに

```

2 3 4  FOR K=1 TO 200
2 3 5  X = (RND (1) +RND (2) +RND (3)) /3 *100 :GOTO 270
を追加し
3 6 0  NEXT K
と変更すると自動的に正規分布ヒストグラムを作る。

```

ディスクを使ったプログラム例

<例1>

*機能

指定した名前のファイルを複写し、新たに指定した名前のファイルを作る。ランダムファイルでもシーケンシャルファイルでもよい。

*ポイント

FIELD文に注意。文字変数の最大長は255バイトなので、FIELD文に1つの文字変数しか定義しない場合、1バイト削られてしまう。

```

10  ' *****
20  ' *
30  ' *          ファイル コピー
40  ' *
50  ' *****
60  PRINT CHR$(12);
70  PRINT " *** ファイル メイ ヲ イレテ クタサイ"
80  INPUT "インプット"; S$
90  INPUT "アウトプット"; D$
100 OPEN S$ AS #1
110 OPEN D$ AS #2
120 FIELD #1, 128 AS S1$, 128 AS S2$
130 FIELD #2, 128 AS D1$, 128 AS D2$
140 IF LOC(1) = LOF(1) THEN 200
150 GET #1
160 LSET D1$ = S1$
170 LSET D2$ = S2$
180 PUT #2
190 GOTO 140
200 PRINT "end"
210 CLOSE
220 END

```

<例2>

*機能

シーケンシャルファイルの名前と調べたい文字列を入力すると、同じ文字列があるレコードの内容を画面に出す。終了するときは、RETURNキーを押す。ファイルの1レコードは、256バイト以下であること。

*ポイント

LINE INPUT文で1レコードを文字変数として読む。


```

10  ' *****
20  ' *                                     *
30  ' *          ストリング   サーチ          *
40  ' *                                     *
50  ' *****
60  INPUT "ファイルメイ"; F$
70  WD$=" "; INPUT "シラベル ストリング "; WD$
80  OPEN F$ FOR INPUT AS #1
90  IF WD$=" " THEN 160
100 ' loop
110   IF EOF(1) THEN 150
120   LINEINPUT#1, L$
130   IF INSTR(L$, WD$) THEN PRINT L$
140   GOTO 100
150  CLOSE#1:GOTO 70
160  CLOSE#1

```

<例3>

*機能

アスキー形式でSAVEされたプログラムA, Bを比較し, 両者の違いを画面に出力する。比較のキーは, 文番号。

*ポイント

キーの大小比較によるシーケンシャルファイルの処理は, この形が基本。

```

10  ' *****
20  ' *          プログラム   ヒカク          *
30  ' *****
40  HV=99999
50  INPUT "file A:"; A$
60  INPUT "file B:"; B$
70  OPEN A$ FOR INPUT AS #1
80  OPEN B$ FOR INPUT AS #2
90  GOSUB 170
100 GOSUB 190
110 IF AK<BK THEN PRINT "A ノミ"; AF$
      :GOSUB 170:GOTO110
120 IF AK>BK THEN PRINT "B ノミ"; BF$
      :GOSUB 190:GOTO110
130 IF AK=HV THEN PRINT "オワリ"
      :CLOSE:END
140 IF AF$<>BF$ THEN PRINT "A ヲ "; AF$
      :PRINT "B ヲ "; BF$
150 GOSUB 170:GOSUB 190
160 GOTO 110
170 IF EOF(1) THEN AK=HV
      ELSE LINEINPUT#1, AF$:AK=VAL(AF$)
180 RETURN
190 IF EOF(2) THEN BK=HV
      ELSE LINEINPUT#2, BF$:BK=VAL(BF$)
200 RETURN

```

<例4>

*機能

カーソルを動かして作った画面 (キャラクターのみ) をディスクに保存する。プリンターに出力することもできる。

*ポイント

画面の内容をGET@で配列に取り込み、ディスクに書き出す。

```
10  ' *****
20  ' *   カ*メン ファイル   (1)   *
30  ' *****
40  WIDTH80, 25:CONSOLE24, 24, 0:COLOR, 32
50  DEFUSR1=&H124A:DIM A%(99), A$(99)
60  PRINT CHR$(12)
70  LOCATE20, 0:PRINT"*** カ*メン ツクリ ***"
80  FOR I=1 TO 2000:NEXT
90  PRINT CHR$(12)
100 LOCATE 0, 24
110 INPUT "オワツタラ RETURN キー";W$
120 LOCATE 0, 24:PRINT
130 INPUT "プ*リント (y/n)";W$:PRINT:PRINT
140 IF W$="y" THEN P=USR1(1)
150 INPUT "シュウセイ (y/n)";W$:PRINT:PRINT
160 IF W$="y" THEN 100
170 INPUT "ファイル (y/n)";W$:PRINT:PRINT
180 IF W$="n" THEN 210
190 INPUT "カ*メン ノ ナマエ";GN$:PRINT:PRINT
200 GOSUB 220
210 END
220 OPEN GN$ AS#1
230 FOR J=0 TO 99
240   FIELD#1, J*2 AS DM$, 2 AS A$(J)
250 NEXT J
260 FOR I=0 TO 79 STEP 4
270   GET@A(I, 0)-(I+3, 24), A%
280   FOR J=0 TO 99
290     LSET A$(J)=MKI$(A%(J))
300   NEXT J
310   PUT#1
320 NEXT I
330 CLOSE#1
340 RETURN
```

<例5>

*機能

例4で作ったファイルを読んで画面に出す。必要ならば修正を加えてディスクに保存する。プリンターに出力することもできる。

*ポイント

ディスク上のファイルを読み、PUT@で画面にだす。

```

10  ' *****
20  ' *   カ`メン ファイル   (2)   *
30  ' *****
40  WIDTH80, 25:CONSOLE24, 24, 0:COLOR, 32
50  DEFUSR1=&H124A:DIM A%(99), A$(99)
60  PRINT CHR$(12)
70  INPUT"カ`メン メイ";GN$:PRINT CHR$(12)
80  GOSUB 210
90  LOCATE0, 24:PRINT
100 INPUT"フ`リント (y/n) ";W$:PRINT:PRINT
110 IF W$="y" THEN PRINT:P=USR1(1)
120 LOCATE0, 24
130 INPUT"シュウセイ シマスカ (y/n) ";W$:PRINT
140 IF W$="n" THEN 170
150 INPUT"オワッタラ RETURN キー ";W$:PRINT
160 GOTO 90
170 INPUT"ファイル シマスカ (y/n) ";W$:PRINT
180 IF W$="n" THEN 200
190 GOSUB 340
200 END
210 OPEN GN$ AS#1
220 FOR J=0 TO 99
230   FIELD#1, J*2 AS DM$, 2 AS A$(J)
240 NEXT J
250 FOR I=0 TO 79 STEP 4
260   GET#1
270   FOR J=0 TO 99
280     A%(J)=CVI(A$(J))
290   NEXT J
300   PUT@A(I, 0)-(I+3, 24), A%
310 NEXT I
320 CLOSE#1
330 RETURN
340 INPUT"カ`メン メイ";GN$:PRINT:PRINT
350 OPEN GN$ AS#1
360 FOR J=0 TO 99
370   FIELD #1, J*2 AS DM$, 2 AS A$(J)
380 NEXT J
390 FOR I=0 TO 79 STEP 4
400   GET@A(I, 0)-(I+3, 24), A%
410   FOR J=0 TO 99
420     LSET A$(J)=MKI$(A%(J))
430   NEXT J
440   PUT#1
450 NEXT I
460 CLOSE#1
470 RETURN

```


<例6>

*機能

本の索引ファイルを作る。必要ならば、作ったファイルを分類してプリンターに出す。入力は、項目名、ページのどちらを先に入れてもよい。ページは頭にPをつけて入れること。ドライブ2にデータファイルを作る。

*ポイント

行番号360は、プログラムのチェイン。

```
10 CLEAR 10000
20 DIM CF$ (101), AA$ (401)
30 OPEN "2:INDEX.DAT" AS#1
40 FIELD#1, 20 AS AF$, 1 AS BF$
50 FOR I=1 TO 101
60     FIELD#1, 54+(I-1)*2 AS DM$,
        2 AS CF$ (I)
70 NEXT I
80 FS=LOF (1)
90 IF FS>0 THEN FOR I=1 TO FS:GET#1, I:
        AA$ (I)=AF$:NEXT I
100 ' LOOP
110 WD$=" ":INPUT WD$
120 IF WD$=" " THEN 340
130 IF NOT ((LEFT$ (WD$, 1) ="P" OR
        LEFT$ (WD$, 1) ="p") AND
        VAL (MID$ (WD$, 2)) >0) THEN 150
140 PG=VAL (MID$ (WD$, 2)):GOTO 330
150 ' ELSE
160 WD$=LEFT$ (WD$+SPACE$ (20), 20)
170 AA$ (FS+1)=WD$
180 FOR I=1 TO FS+1
190     IF AA$ (I)=WD$ THEN 210
200 NEXT I
210 IF NOT (I>FS) THEN 260
220 FS=FS+1
230 LSET AF$=WD$:LSET BF$=CHR$ (1)
240 LSET CF$ (1)=MKI$ (PG)
250 PUT#1, FS:GOTO 330
260 ' ELSE
270 GET#1, I:B=ASC (BF$)
280 LSET CF$ (B+1)=MKI$ (PG)
290 FOR J=1 TO B+1
300     IF CVI (CF$ (J))=PG THEN 320
310 NEXT J
320 IF J>B THEN LSET BF$=CHR$ (J):
        PUT#1, I
330 GOTO 100
340 CLOSE#1
350 INPUT "フ リント シマスカ (y/n) ";YN$
360 IF YN$="y" THEN RUN "pinx"
```

<例7>

*機能

例6で作ったファイルを読み、分類してプリンターへ出力する。データを追加することもできる。ドライブ2のデータファイルを読む。

*ポイント

行番号60~140, 150~260でそれぞれ項目、ページの分類をする。

```

10  CLEAR 10000: DIM CF$ (101), C% (101),
      AA$ (401), RN% (401)
20  OPEN "2: INDEX. DAT" AS#1
30  FIELD#1, 20 AS AF$, 1 AS BF$
40  FOR I=1 TO 101: FIELD#1, 54 + (I-1) * 2
      AS DM$, 2 AS CF$ (I): NEXT I
50  FS=LOF (1): IF NOT (FS>0) THEN 300
60  FOR I=1 TO FS: GET#1, I: L=1: R=I-1
70  ' LOOP
80  IF L>R THEN 120
90  M= (L+R) ¥2
100  IF AA$ (M) > AF$ THEN R=M-1
      ELSE L=M+1
110  GOTO 70
120  IF L<=I-1 THEN
      FOR J=I-1 TO L STEP -1:
      AA$ (J+1) = AA$ (J):
      RN% (J+1) = RN% (J): NEXT J
130  AA$ (L) = AF$: RN% (L) = I
140  NEXT I
150  FOR I=1 TO FS
160  GET#1, RN% (I): B=ASC (BF$)
170  FOR J=1 TO B
180  C=CVI (CF$ (J)): L=1: R=J-1
190  ' LOOP
200  IF L>R THEN 240
210  M= (L+R) ¥2
220  IF C% (M) > C THEN R=M-1
      ELSE L=M+1
230  GOTO 190
240  IF L<=J-1 THEN
      FOR K=J-1 TO L STEP -1:
      C% (K+1) = C% (K): NEXT K
250  C% (L) = C
260  NEXT J
270  LPRINT USING
      "& " ; AF$ ;
280  FOR J=1 TO B: LPRINT USING
      "####" ; C% (J) ; : NEXT J: LPRINT
290  NEXT I
300  CLOSE#1
310  INPUT "マタ トウロクシマスカ (y/n) " ; YN$
320  IF YN$ = "y" THEN RUN "inx"

```

<例8>

*機能

社員コードの順番に売り上げを入力し、ハンバイファイルを作る。次のプログラムでは、ハンバイファイルを読み、報告書を作る。報告書は、社員コードで売り上げ金額の小計、社員コードの頭1桁で中計、すべての社員の大計をとる。

*ポイント

集計をとるプログラムは、この形が基本。報告書作成プログラムの行番号90, 100で行っているキーが変わったときの処理に注意。

入力データ

社員コード	売り上げ金額
111	5,000
111	4,000
115	2,500
200	1,500
250	2,000
250	4,500
360	5,500
410	3,000
410	7,000
450	3,500

出力結果

111	5,000		
111	4,000		
		111ケイ	9,000
115	2,500		
		115ケイ	2,500
<hr/>			
		1ショウケイ	11,500
<hr/>			
200	1,500		
		200ケイ	1,500
250	2,000		
250	4,500		
		250ケイ	6,500
<hr/>			
		2ショウケイ	8,000
<hr/>			
360	5,500		
		360ケイ	5,500
<hr/>			
		3ショウケイ	5,500
<hr/>			
410	3,000		
410	7,000		
		410ケイ	10,000
450	3,500		
		450ケイ	3,500
<hr/>			
		4ショウケイ	13,500
<hr/>			
		ソウゴウケイ	38,500
<hr/>			


```

10 ' *****
20 ' * データ ニュウリョク *
30 ' *****
40 OPEN "ハンパイ" FOR OUTPUT AS #1
50 INPUT "シャインコート"; CD$
60 IF CD$="999" THEN GOTO 100
70 INPUT "ウリアケ"; UR
80 PRINT #1, CD$; ", "; UR
90 GOTO 50
100 CLOSE#1
110 END

```

```

10 ' *****
20 ' * ホウコクショ サクセイ (シュウケイ ノ シカタ) *
30 ' *****
40 OPEN "ハンパイ" FOR INPUT AS #1
50 T1=0:T2=0:T3=0:SW=0
60 IF EOF(1) THEN GOTO 140
70 INPUT#1, CD$, UR
80 IF SW=0 THEN SW=1:SV$=CD$
90 IF LEFT$(CD$, 1) <> LEFT$(SV$, 1) THEN
    GOSUB 170:GOSUB 220:SV$=CD$
100 IF CD$<>SV$ THEN GOSUB 170:SV$=CD$
110 LPRINT USING "& & ###, ###"; CD$; UR
120 T1=T1+UR
130 GOTO 60
140 GOSUB 170:GOSUB 220:GOSUB 300
150 CLOSE#1
160 END
170 ' ショウケイ ノ フ リント
180 LPRINT TAB(12); SV$; "ケイ";
190 LPRINT USING "###, ###"; T1
200 T2=T2+T1:T1=0
210 RETURN
220 ' チュウケイ ノ フ リント
230 LPRINT STRING$(30, "-")
240 LPRINT TAB(11); LEFT$(SV$, 1);
250 LPRINT "ショウケイ";
260 LPRINT USING "###, ###"; T2
270 LPRINT STRING$(30, "-")
280 T3=T3+T2:T2=0
290 RETURN
300 ' タ イケイ ノ フ リント
310 LPRINT TAB(10); "ソウコウケイ";
320 LPRINT USING "###, ###"; T3
330 LPRINT STRING$(30, "-")
340 RETURN

```

PC-6001の特長

- (1) PC-6001のBASICもPC-8001と同様にマイクロソフト社のBASICを土台にしている。しかし、PC-8001と内部コードは異なる。
- (2) キャラクターコードも「ひらがな」を追加したことにより、PC-8001とは異なる。
- (3) 音楽機能が追加された。SOUND, PLAYを見よ。グラフィック機能が強化された。PAINT, COLOR, LINEなどを見よ。
- (4) 複数のページを持っているので、テキスト画面とグラフィック画面などを別々に持つことができる。SCREENを見よ。
- (5) ROMカートリッジ、ROM&RAMカートリッジを装着できる。ゲーム用にジョイスティックが2個つけられる。ディジタイザーも使える。オーディオ出力が可能になった。
- (6) 家庭用テレビに直接つなぐことができる。アダプターは必要ない。
- (7) オプションでRS-232Cボードが用意されている。
- (8) キーボードにオーバーレイシートを使用した。標準では、ひらがなのオーバーレイシートがついている。グラフィックやカタカナのシートと取り替え可能。

PC-6001の扱い方

- (1) PC-6001に接続されている機器の電源を入れた後にPC-6001の電源を入れる。
- (2) PC-6001の画面に How Many Pages? と表示されるので、使用する画面のページ数を入れる。ROM&RAMカートリッジがある場合4ページ、ない場合2ページが最大ページ数。ページ数の指定をせずにRETURNキー押すと最大ページ数がとられる。
- (3) テレビにつないだときは、チャンネルを1または2（その地域で使用していないチャンネル）にする。PC-6001のチャンネルは、RF出力端子の下にある青いスイッチで切り換える。
- (4) ROM&RAMカートリッジを使用するときは、カートリッジ挿入孔に入れる。PC-6001の電源が入っている状態で挿入すると、PC-6001の電源が1度切れた後で再び入る。ROMのプログラムを実行するときは、この後でRETURNキーを押す。ROM&RAMカートリッジを使用しているときにプログラムをカセットテープに保存した場合には、プログラムを読み込むときにも必ずROM&RAMカートリッジをつけておくこと。

PC-6001 キー配置図

ファンクションキーの定義
 SHIFT+F1→SCREEN F1のみ→COLOR
 SHIFT+F2→CSAVE F2のみ→CLOAD
 SHIFT+F3→PRINT F3のみ→GOTO

ファンクションキー→		F 1		F 2		F 3		F 4
ESC	!	”	#	\$	%	&	'	(
	1	2	3	4	5	6	7	8
TAB		Q	W	E	R	T	Y	U
CTRL		A	S	D	F	G	H	J
SHIFT		Z	X	C	V	B	N	
(1) キー			GRAPH		ス ペ ー ス キ ー			
①								

キーを押すと、③の位置の文字が出る。英字の場合は②の位置の小文字が出る。SHIFT を押してキーを押すと①の位置の文字が出る。英字の場合は②の位置の大文字が出る。

(2) TABキー
 8カラムごとにカーソルを右へ移動する。

(3) ESCキー
 プログラムの一時停止。再開はキーボード上のどのキーを押してもよい。

(4) CTRLキー
 スクリーンエディットのときに使う。このキーを押しながら次のキーを押す。

B→カーソルを1項目ごとにも左にずらす。	J→ラインフィード。
C→STOPキーと同じ。	K→HOMEキーと同じ。
E→カーソルから行の終わりまでを消す。	L→CLRキーと同じ。
F→カーソルを1項目ごとにも右にずらす。	M→RETURNキーと同じ。
G→BEEPと同じ。	R→INSキーと同じ。
H→1文字削除。	U→カーソルのある行を1行抹消する。
I→TABキーと同じ。	

(5) STOPキー
プログラムの強制停止。

(6) CLR → 押すと画面クリアー。
HOME → SHIFT+CLR カースルが左上へ。
INS → 押すとカースルの左に文字等を挿入で
きる。解除はINS, RETURN, DEL, カース
ル移動のいずれかのキーを押す。
DEL → 押すとカースルの左1文字を削除。

SHIFT+F4 → PLAY SHIFT+F5 → CONT+RETURN
F4のみ → LIST F5のみ → RUN+RETURN

F 4		F 5		HOME CLR		STOP	INS DEL	
)	9	=	0	-	^	¥	←	↵
←画面切り替えキー カースル移動。 ↓								
I	O	P	@	{	RETURN			
K	L	+	*	;	:)	か	な
M	<	>	?	/	—	SHIFT		
						↑		
						←	→	
						↓		

(7) RETURNキー
入力の終わりに必ず押すこと。
(8) スペースキー
このキーを押すと空白がひとつとられる。

(9) かなキー

押すとロックされてオーバーレイシートに書かれているひらがなになる。解除はもう一度押す。SHIFT キーを押しながら画面切り替えキーを押すとカタカナに。

(10) 画面切り替えキー

画面表示するページを切り替える。画面の左下に表示画面のページ数が出る。

												STOP		
	日	月	火	水	木	金	土	百	千	万	—	円		
				「	〒	」	年			π		○	RETURN	
			秒	ト	+	+	時			中	♣	♥	●	
			×	└	┐	┘		分	小	大	♠	◆		
GRAPH → キー														

上の文字および図形を出すには、GRAPH キーを押しながら各キーを押す。

PC-6001 文字・演算子

機能	&H 疑問符 (?) コロン (:) セミコロン (;) コンマ (,) 引用符 (")
型宣言	\$
算術演算	= - + * / ^
論理演算	NOT AND OR
関係演算	= < > <= =< >= => <> ><

PC-6001 コマンド・ステートメント一覧表

コマンド	LIST LLIST NEW CONT RUN CLOAD CLOAD? CSAVE EXEC CLEAR
ステートメント	REM END FOR~NEXT GOSUB GOTO ON~GOTO ON~GOSUB RETURN STOP IF~THEN LET DIM DATA READ RESTORE POKE SOUND PLAY DEFFN IF~GOTO
入出力ステートメント	INPUT PRINT LPRINT INPUT# PRINT# LCOPY OUT
画面ステートメント	CONSOLE COLOR CLS LOCATE PSET PRESET LINE PAINT SCREEN
特殊ステートメント	KEY
数値関数	ABS COS EXP INT LOG RND SGN SIN SQR TAN
ストリング関数	CHR\$ LEFT\$ MID\$ RIGHT\$ STR\$ ASC LEN VAL INKEY\$
入出力関数	INP STICK STRIG
その他の関数	CSRLIN POS LPOS PEEK FRE SPC TAB TIME POINT USR

PC-8001と機能がまったく同じものについては、説明を省略した。1部機能が異なるものについては、その部分だけ説明した。

プログラムの1行に認められる命令の長さおよびINPUT文で入力できるデータの長さは、71文字まで。

数値は有効桁数9桁。数値の範囲は、 3×10^{-38} から 1.7×10^{38} まで。文字列に入る文字の数は、255文字まで。

(1) コマンド

LIST [<開始行番号>] [- [<終了行番号>]]

15ページ参照。LISTは使えない。

CLOAD "<ファイル名>"

14ページ参照。ファイル名を指定しないと最初に見つけたプログラムを読み込む。

EXEC<番地>

指定した番地からの機械語プログラムを実行する。

EXEC&HD100-----D100番地からの機械語プログラムを実行する。

機械語プログラム実行後BASICに戻るためには、機械語プログラムの最後をRET(16進のC9)とする。機械語プログラムの書き込みは、POKEで行う。

(2) ステートメント

IF<条件式>THEN<文または行番号>

21ページ参照。ELSEは使えない。

DEFN (<引数>)

19ページ参照。引数は1種類しか使えない。

SOUND <レジスター番号>, <データ>

サウンド発生用ICを直接制御する。レジスター番号でIC内部のレジスターを指定し、データを書き込む。詳細は、172ページ参照。

PLAY<ボイスA文字列> [, <ボイスB文字列>] [, <ボイスC文字列>]

音楽を発生する。3音までの和音が出せる。

文字列には、ミュージック・マクロ・ランゲージと呼ばれる音楽演奏のための記号を書く。詳細は、175ページ参照。

CONSOLE [<スクロール開始行>], [<スクロールの長さ>],
[<ファンクションキー表示>], [<キークリック音>]

画面モード1, 2の場合のテキスト画面の制御をする。

スクロール開始行	0から最終行まで。最終行は、ファンクションキーを表示しているとき14, 表示していないとき15。これより大きい値を使うと、最終行からとみなされる。
スクロールの長さ	1から最終行まで。最終行より大きいと最終行まで。
ファンクションキー表示	0なら表示しない。1なら表示する。
キークリック音	0ならキーイン音なし。1ならキーインの音あり。

COLOR [<文字やグラフィックスの色>] [, <背景色>]
[, <色の組み合わせ>]

画面の色を指定する。詳細は、176ページ参照。

CLS

アクティブページ(文字や図形が描かれているページ)を消す。

LOCATE X, Y

28ページ参照。カーソルを消すことはできない。

PSET (X, Y) [, <色>]

画面上の座標(X, Y)の位置に点を描く。X, Yの範囲はモードによらない。Xは0~255, Yは0~191。左上が(0, 0)である。
色を指定しないとフォアグラウンドカラーになる。176ページ参照。

PRESET (X, Y)

PSETで描いた点を消す。バックグラウンドカラーでPSETするのと同じ。

LINE [(A, B)] - (C, D) [, <色>] [, <BまたはBF>]

はじめの座標を省略すると、直前に実行したLINE, PSET, PRESET, POINTで指定した座標からの意味となる。なければ(0, 0)からとなる。色は、176ページ参照。B, BFはPC-8001と同じ。

PAINT (X, Y) [, <色> [, <境界色>]]

指定した点から境界色で囲まれた範囲を指定した色で塗りつぶす。色は、176ページ参照。

SCREEN [<モード>] [, <アクティブページ>]
[, <ビジュアルページ>]

画面モード	1	文字モード	
	2	64×48グラフィックス, 文字	
	3	128×192グラフィックス	
	4	256×192グラフィックス	
アクティブページ (文字や図形を描く)	1	ページ1に描く	
	2	ページ2に描く	
	3	ページ3に描く	(拡張RAMが必要)
	4	ページ4に描く	(拡張RAMが必要)
ビジュアルページ (画面に表示する)	1	ページ1を表示する	
	2	ページ2を表示する	
	3	ページ3を表示する	(拡張RAMが必要)
	4	ページ4を表示する	(拡張RAMが必要)

ページ1はモード1と2が可能。ページ2, 3, 4はモード1, 2, 3, 4が可能。

(3) 入出力ステートメント

INPUT#-<入力機器番号>, <変数> [, <変数>]

入力機器番号#-0	キーボード
#-1	カセットテープレコーダー
#-2	RS-232C

PRINT#-<出力機器番号>, <式> [, <式>]

出力機器番号#-0	ディスプレイ
#-1	カセットテープレコーダー
#-2	RS-232C
#-3	プリンター

LCOPY

画面と同じ絵柄を専用プリンターに出力する。

(4) 一般関数

POINT (X, Y)

指定した位置の点の色コードが求められる。176ページ参照。

TIME

PC-6001内部の時間の経過を表す数値をあたえる。数値は、 $(1/1024)$ 秒ごとに1増える。PC-6001内部の1秒は、 $(1000/1024=0.9765625)$ 秒なので、正しい1秒にするには1.024を掛ける必要がある。これでもハードの影響をうけて誤差がでることに注意。

USR (X)

機械語プログラムを実行する。Xは機械語プログラムに渡す値。BASIC側では、機械語プログラムの開始番地を&HFAEB番地と&HFAEC番地に、POKE文で下位バイト、上位バイトの順に書き込む。機械語プログラムでは、741H番地をコールすることにより、データXの値をDEレジスターに得ることができる。Xの値は、-32768~32767の間。機械語プログラムからBASICプログラムにデータを返すには、ABレジスターにデータを入れてD16H番地にジャンプする。

(例) 任意の整数に5を足す (PC-8001の例も参照せよ。48ページ)

```
10 CLEAR 50, &HCFFF
20 FOR J=0 TO 11: READ M
30 POKE &HD000+J, M: NEXT J
40 DATA &HCD, &H41, &H07, &H21, &H05, &H00
50 DATA &H19, &H7C, &H45, &HC3, &H16, &H0D
60 POKE &HFAEB, &H00: POKE &HFAEC, &HD0
70 INPUT "データは"; R
80 W=USR (R)
90 PRINT W
```

機械語部分をニーマニックで書くと
右のようになる。変換表参照。

```
CALL 0741H
LD HL, 5
ADD HL, DE
LD A, H
LD B, L
JP 0D16H
```


(5) 入出力関数

STICK (n)

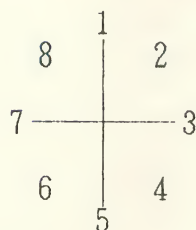
ジョイスティックの状態をあたえる。

n=0 キーボードカーソルキー

n=1 ジョイスティック 1

n=2 ジョイスティック 2

カーソルキーを押した状態、ジョイスティックをたおした状態は、下の数値と対応している。



カーソルキーで 2・4・6・8 など斜め方向を表すには 2 個同時にキーを押す。
たとえば、6 は ← キー と ↓ キー を同時に押す。

STRIG (n)

ジョイスティックのトリガーボタンの状態、キーボードのスペースキーの状態をあたえる。押されていれば 1、はなされていれば 0 を示す。

n=0 キーボードスペースキー

n=1 ジョイスティック 1 ボタン

n=2 ジョイスティック 2 ボタン

(例)

```
10 PRINT STRIG (1) : GOTO 10
```

ジョイスティック 1 のトリガーボタンが押されていれば 1、はなされていれば 0 を表示し続ける。

(6) スtring関数

CHR\$ (X)

80 ページ参照。例外に以下のものがある。

CHR\$ (5)	CTRL キー + E
CHR\$ (8)	DEL キー
CHR\$ (12)	CLR キー
CHR\$ (13)	RETURN キー
CHR\$ (18)	INS キー
CHR\$ (21)	CTRL キー + U
CHR\$ (34)	ダブルクォーテーション (")

各機能の詳細

SOUND <レジスター番号>, <データ>

*レジスターの表

レジスター	機 能	データの範囲	
0	ボイス1の高さ (細かく変える)	0 ~ 255	
1	ボイス1の高さ (粗く変える)	0 ~ 15	
2	ボイス2の高さ (細かく変える)	0 ~ 255	
3	ボイス2の高さ (粗く変える)	0 ~ 15	
4	ボイス3の高さ (細かく変える)	0 ~ 255	
5	ボイス3の高さ (粗く変える)	0 ~ 15	
6	ノイズの高さ	0 ~ 31	
7	ノイズ, ボイスのオン・オフ	0 ~ 63	
8	第1音の大きさ	0 ~ 15	16ならトレモロ
9	第2音の大きさ	0 ~ 15	16ならトレモロ
10	第3音の大きさ	0 ~ 15	16ならトレモロ
11	トレモロの早さ (細かく変える)	0 ~ 255	
12	トレモロの早さ (粗く変える)	0 ~ 255	
13	トレモロの形 (エンベロープ)	0 ~ 15	

*音を出す時は次のように考えればよい。

1. ボイスを出すか, ノイズを出すかをきめる。ボイスとは, きれいな音のこと。ノイズとは, シューあるいはザーという音のこと。混ぜることもできる。
2. ボイス, ノイズそれぞれの音の高さをきめる。
3. 音の大きさをきめる。エンベロープを使うときは, 16をセットする。
4. エンベロープの早さと形をきめる。

*レジスター {0, 1}, {2, 3}, {4, 5} に入れる値

それぞれ2つのレジスターを1組として, 各組には各ボイスの周波数を決定する数値を入れる。数値は次の式によって計算する。

$$TP = f_{clock} / (16 \times fT) \quad CT + FT / 256 = TP / 256$$

f c l o c k : 1. 99675MHz

f T : 出力する周波数

FT : レジスター0, 2, 4いずれかの値

CT : レジスター1, 3, 5いずれかの値

ボイス1の出力する周波数を100Hzとすると

$$TP = 1.99675 \times 10^6 / (16 \times (1 \times 10^2)) \approx 1248$$

$$CT + FT / 256 = 1248 / 256 = 4 + 224 / 256$$

よって

$FT = 224$ $CT = 4$ (いずれも10進数)
と設定すればよい。

*レジスター6に入れる値

ノイズの周波数を決定する数値を入れる。数値は次の式によって計算する。

$$NP = \text{f c l o c k} / (16 \times fN)$$

f c l o c k : 1. 99675MHz

f N : 出力するノイズ周波数

NP : レジスター6に設定する数値

出力するノイズ周波数を10KHzとすると

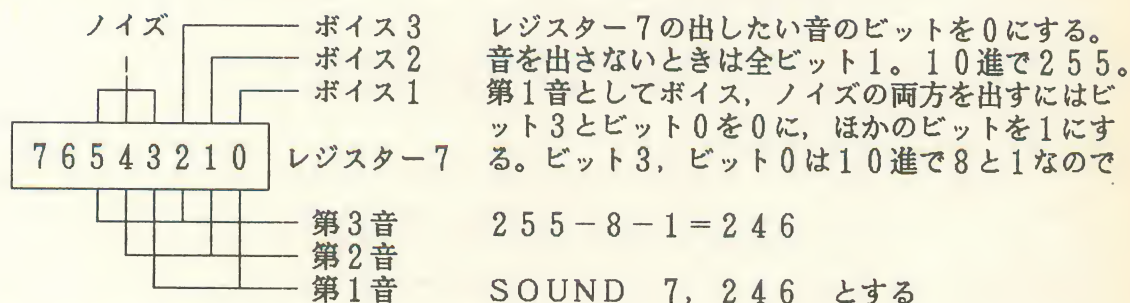
$$NP = 1.99675 \times 10^6 / (16 \times (10 \times 10^3)) \approx 12$$

よってレジスター6には、12(10進数)を設定すればよい。

*レジスター7に入れる値

サウンド発生用ICは3つのボイスと1つのノイズをもっている。

レジスター7はこれらをオン・オフすることによって合成音をつくる。



*レジスター8, 9, 10に入れる値

各音の大きさを決定する値を入れる。0~15の値なら音の大きさは一定。0なら無音。16なら音の大きさが変化するモード(エンベロープモード)となる。

*レジスター11, 12に入れる値

2つのレジスターにエンベロープの周期を決定する数値を入れる。数値は、次の式によって計算する。

$$EP = \text{f c l o c k} / (256 \times fE) \quad CT + FT / 256 = EP / 256$$

f c l o c k : 1. 99675MHz

f E : 音が増加, 減少する周波数

FT : レジスター11に設定する数値

CT : レジスター12に設定する数値

音が減衰して0になるまでの時間を0.5秒に設定すると

$$f_E = 1 / 0.5 = 2 \text{ Hz}$$

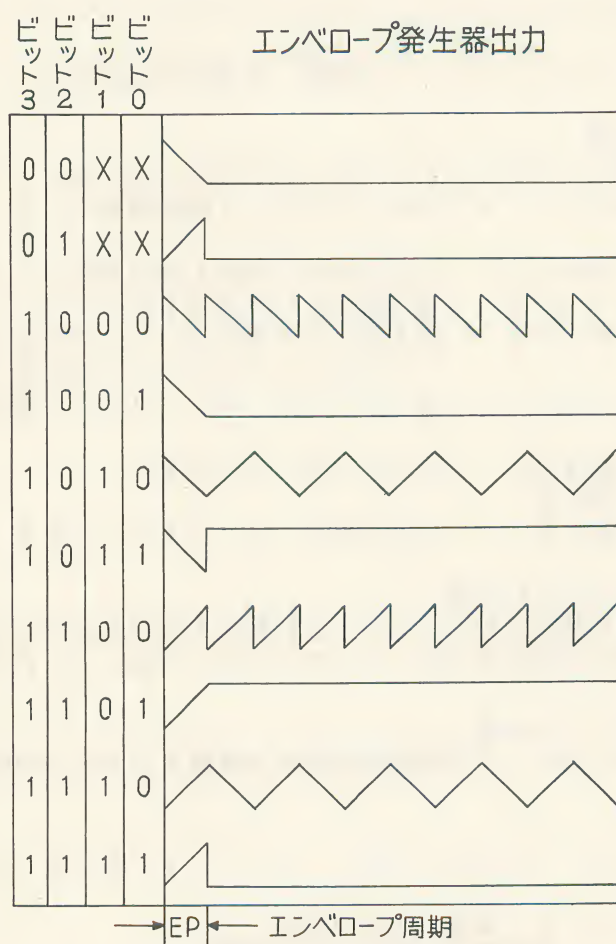
$$EP = 1.99675 \times 10^6 / (256 \times 2) \approx 3900$$

$$CT + FT / 256 = 3900 / 256 = 15 + 60 / 256$$

よって $FT = 60$ $CT = 15$ と設定する。

*レジスタ11, 12に入れる値

エンベロープの形状を決定する値を入れる。レジスタ13のビット0~3の値が次のような波形と対応する。xは0でも1でも影響ない。



PLAY<ボイスA文字列>〔, <ボイスB文字列>〕〔, <ボイスC文字列>〕

ミュージック・マクロ・ランゲージ (以下, MMLと書く) の書きかた

*音程

C	D	E	F	G	A	B
ド	レ	ミ	ファ	ソ	ラ	シ

半音は, 後ろに#なら+を, bなら-をつける

*音域 (オクターブ)

O n nは1~8 オクターブを変える。はじめはO 4になっている。

*長さ

音符の長さを数字でしめす。4分音符なら4, 16分音符なら16。
符点付きの音符は, ピリオドを符点の数だけつける。

O n X〔<+または->〕m〔.〔.〕〕

↑ 符点 ひとつ付けると半分ずつのびる
↑ 長さ 4分音符なら 4
↑ 半音 シャープなら+, フラットなら-
↑ 音程 ドなら C
↑ オクターブ

*休符

R n nは1~64 休符。

*各文字の意味

M x	エンベロープ周期の設定 (xは1~6 5 5 3 5)	
S x	エンベロープ形状の設定 (xは0~1 5)	
V x	音量の設定 (xは0~1 5)	指定しないときは 8
L x	長さの設定 (xは1~6 4)	指定しないときは 4
O x	オクターブの設定 (xは1~8)	指定しないときは 4
T x	テンポの設定 (xは3 2~2 5 5)	指定しないときは 1 2 0
R x	休符の設定 (xは1~6 4)	指定しないときは 4
X +	音を半音上げる	
X -	音を半音下げる	
N x	数値で音程を指定する (xは1~9 6)	

*注意点

- ・ Lでは長さを指定しない音符の長さを決める。いちばん多く使うものにしておけば, そのつど長さを指定しなくてもすむので便利。休符の長さは変わらない。
- ・ M, SとVとはどちらか一方しか指定できない。
- ・ Nで音程を指定するときは, 音程を表すxと長さを表すxとの間をセミコロンでくぎる。Nでだせる最低音N 1は, O 1 C+と同じ高さ。xが1増えるごとに半音ずつ上がり, 9 6ではO 8 Bと同じ高さ。

<モード3>

F	B	C	文字またはグラフィックスの色
1	1~4	1	緑
2	1~4	1	黄
3	1~4	1	青
4	1~4	1	赤
1	1~4	2	白
2	1~4	2	水色 (シアン)
3	1~4	2	紫 (マゼンタ)
4	1~4	2	だいだい

<モード4>

F	B	C	文字またはグラフィックスの色
1	0	1	黒地に緑
0	1	1	緑地に黒
1	0	2	黒地に白
0	1	2	白地に黒

SCREEN 2,2,2:COLOR 2,0,1
 SCREEN 7,2,2:COLOR 1,0,2

キP
ャC
ラ6
ク0
タ0
ー1
・
コ
ード
表

1616
進進
数数
には
対横
応が
する
位
104
進ビ
ット
た縦
とが
え下
ば位
4B4
はビ
ット
だが
10マ
進ス
では
中75
の数
字は

16進	0	1	2	3	4	5	6	7
0	0 0	π 16	32	0 48	@ 64	P 80	96	p 112
1	月 1	\perp 17	! 33	1 49	A 65	Q 81	a 97	q 113
2	火 2	T 18	" 34	2 50	B 66	R 82	b 98	r 114
3	水 3	十 19	# 35	3 51	C 67	S 83	c 99	s 115
4	木 4	ト 20	\$ 36	4 52	D 68	T 84	d 100	t 116
5	金 5	十 21	% 37	5 53	E 69	U 85	e 101	u 117
6	土 6	 22	& 38	6 54	F 70	V 86	f 102	v 118
7	日 7	一 23	' 39	7 55	G 71	W 87	g 103	w 119
8	年 8	「 24	(40	8 56	H 72	X 88	h 104	x 120
9	円 9	ㄣ 25) 41	9 57	I 73	Y 89	i 105	y 121
A	時 10	L 26	* 42	: 58	J 74	Z 90	j 106	z 122
B	分 11	┘ 27	+ 43	; 59	K 75	(91	k 107	{ 123
C	秒 12	× 28	, 44	< 60	L 76	≡ 92	l 108	 124
D	百 13	大 29	— 45	= 61	M 77) 93	m 109	} 125
E	千 14	中 30	. 46	> 62	N 78	^ 94	n 110	~ 126
F	万 15	小 31	/ 47	? 63	O 79	— 95	o 111	

8 9 A B C D E F 16進

♠ 128			— 176	タ 192	ミ 208	た 224	み 240	0
♥ 129	あ 145	。 161	ア 177	チ 193	ム 209	ち 225	む 241	1
♣ 130	い 146	「 162	イ 178	ツ 194	メ 210	つ 226	め 242	2
♦ 131	う 147	」 163	ウ 179	テ 195	モ 211	て 227	も 243	3
○ 132	え 148	、 164	エ 180	ト 196	ヤ 212	と 228	や 244	4
● 133	お 149	・ 165	オ 181	ナ 197	ユ 213	な 229	ゆ 245	5
を 134	か 150	ヲ 166	カ 182	ニ 198	ヨ 214	に 230	よ 246	6
ぁ 135	き 151	ァ 167	キ 183	ヌ 199	ラ 215	ぬ 231	ら 247	7
い 136	く 152	ィ 168	ク 184	ネ 200	リ 216	ね 232	り 248	8
う 137	け 153	ゥ 169	ケ 185	ノ 201	ル 217	の 233	る 249	9
え 138	こ 154	ェ 170	コ 186	ハ 202	レ 218	は 234	れ 250	A
お 139	さ 155	ォ 171	サ 187	ヒ 203	ロ 219	ひ 235	ろ 251	B
や 140	し 156	ャ 172	シ 188	フ 204	ワ 220	ふ 236	わ 252	C
ゅ 141	す 157	ュ 173	ス 189	ヘ 205	ン 221	へ 237	ん 253	D
よ 142	せ 158	ョ 174	セ 190	ホ 206	・ 222	ほ 238	254	E
っ 143	そ 159	ッ 175	ソ 191	マ 207	・ 223	ま 239	255	F

PC-6001 メモリーアドレスマップ

RAM 16 Kバイトの場合
(スクリーンページが2 ページ)

16進番地

FFFF	BASIC SYSTEM
EA00	ページ 2
E000	プログラム
C400	ページ 1
C000	
8000	
7000	キャラクタージェネレーター (バンク切り替え)
6000	
4000	BASIC インタープリター領域
0000	

拡張
ROM
エリア

RAM 32 Kバイトの場合
(スクリーンページが4 ページ)

16進番地

FFFF	BASIC SYSTEM
EA00	ページ 2
E000	ページ 3
C000	ページ 4
A000	プログラム
8400	ページ 1
8000	
7000	キャラクタージェネレーター (バンク切り替え)
6000	
4000	BASIC インタープリター領域
0000	

拡張
RAM
エリア

拡張
ROM
エリア

RAMを16 Kバイト拡張する場合
ROM&RAMカートリッジを接続する。

ROMを16 Kバイト拡張する場合
ROM&RAMカートリッジまたはROMカートリッジを接続する。

- ?BS Error (Bad subscript Error)
配列の添字がDIMで宣言したものより大きい。
添字の個数が正しくない。
- ?CN Error (Continue Error)
CONT命令を実行できない。プログラムを変更したあとでCONT命令は実行できない。
- ?DD Error (Duplicate Definition Error)
同じ名前の配列をもう1度定義した。
- ?FC Error (Function Call Error)
許容範囲外の数字を使った。
- ?ID Error (Illegal Direct Error)
ダイレクトモードでは使えないコマンドを使った。
INPUT文、DEFBN文など。
- ?LS Error (Long String Error)
文字列が長すぎる。255文字が限度。
- ?MO Error (Missing Operand Error)
不完全な記述がある。代入命令の右辺がないなど。
- ?NF Error (Next without For Error)
FORとNEXTが対応していない。NEXTがあるのにFORがない。
- ?OD Error (Out of Data Error)
READ命令を実行したが、DATAがない。
- ?OM Error (Out of Memory Error)
プログラムが大きすぎてメモリー不足になった。
サブルーチンの中で自分自身のサブルーチンを呼んだ。
- ?OS Error (Out of String Space Error)
文字列用の領域が不足。CLEAR文で増やすこと。
- ?OV Error (Overflow Error)
計算結果が許容範囲を越えた。大きすぎるか、小さすぎる。
- ?RG Error (Return without GOSUB Error)
GOSUB命令がないのにRETURNがある。

?SN Error (Syntax Error)
文法違反。

?ST Error (String too Complex Error)
文字式が複雑すぎる。

?TM Error (Type Mismatch Error)
文字や数値の扱いかが正しくない。数値変数に文字を入れるなど。

?TR Error (Tape Read Error)
テープが正しく読み込めない。

?UF Error (Undefined Function Error)
DEF FN で定義されていないユーザー関数を呼んだ。

?UL Error (Undefined Line number Error)
指定した行番号がない。

?/0 Error (Division by Zero Error)
0で割り算している。

PC-6001シリーズ ラインアップ

型 名	品 名	特 長
PC-6001	PC-6000シリーズのパーソナルコンピュータ	
PC-6005	ROMカートリッジ	2K, 4K, 8KバイトのマスクROM, 8KバイトのPROMを使うことができる
PC-6006	ROM&RAMカートリッジ	RAM16Kバイト ROMは上と同じ
PC-6021	40桁専用サーマルプリンター	画面プリントを取ることができる
PC-6031	ミニフロッピーディスクユニット	
PC-6041	12インチ・グリーンディスプレイ	
PC-6042	12インチ・カラーディスプレイ	
PC-6051	タッチパネル (ディジタイザー)	パネル上の指定した点の位置を高精度で読み取ることができる
PC-6061	RS-232Cボード	300, 600, 1200, 2400, 4800ボアの選択が可能
PC-6071	フィーダー切り替えスイッチ	
PC-6072	キーボード・オーバーレイシート	無地
PC-6073	同上	ひらがな用
PC-6081	データレコーダー	パーソナルコンピュータ用カセットテープレコーダー
PC-6082	データレコーダー	同上。プログラムの頭出しが可能

& (アンパーサント)	10	アスキー形式 (ファイルの属性)	58
&H (16進数)	10	アセンブラ	99
&H81 (129指数のゲタ)	44	頭に¥をつける	42
&H8021 (プログラム開始番地)	86	アトリビュート (属性)	26, 29
&HC021 (同上)	86	アトリビュートエリア	88, 89
&O (8進数)	10	アドレス (番地)	82
' (アポストロフィー)	10	アドレスバス	82
. (ピリオド)	10	アポストロフィー	10
: (コロロン)	10	アンダーライン	53
; (セミコロロン)	10	一覧表示	15
? (疑問符)	10	イネーブル・インタラプト	109
, (コンマ)	10	色や機能の指定	27
" (引用符)	10	インクリメント	105
\$ (文字列宣言)	10	インタフェース	91
% (整数宣言)	10	インタープリター	86
! (単精度宣言)	10	インデックス検出孔	56
# (倍精度宣言)	10	インデックス・レジスター	84, 102
= (代入, 等価判定)	10	105, 127, 130, 132	
- (減算, 負)	10	インプットアドレス	33
+	10	インプリケーション	11
*	10	引用符	10
/ (除算)	10	ウォームスタート	99
^ (べき乗)	10	うずまきを描く	150
¥ (除算・結果整数)	10	エラー	40
< (大小比較)	11	エラーコード	69
> (大小比較)	11	エラー発生シミュレート	31
<> (等しくない)	11	エラーメッセージ	69, 181
=< (以上)	11	エリート	53
>= (以下)	11	演算の優先順位	9
10進数を2進数にする	152	円を描く	150
16進数文字列 (HEX \$)	36	エンベロープ	174
16ビット転送・演算	133	オーバーフロー	83
1歩後退2歩前進 (図形)	145	オープン	59
1文字だけをプリント	42	オープン状態	23
1文字をコードにコードを文字に	152	オペコード	82
2重添字つき変数	145	オペコードフィールド	104
2進・10進・16進	152	折れ線グラフ	153
2進数	44	音響カプラー	90
2の補数	108	音声認識ボード	76
2バイトの文字列	39	カーソル移動	28
3桁ごとにコンマで区切る	43	カーソルの縦の位置	40
4バイトの文字列に変換	39	カーソルの横の位置	41
8進数文字列	37	改行	55
8バイトの文字列に変換	39	改行幅	54
8ビット転送演算	131	回転	136
アークタンジェント (ATN)	34	下位バイト	86
アイテム (項目)	61	書き込み禁止	58

< 索

引 >

書き込む	25
かくしビット (ヒドンビット)	45
拡大文字	53
拡張 I/Oバス	75
拡張 RAMエリア	85
拡張 ROMエリア	85
拡張ユニット	56, 73, 75
加減算フラグ	83
加算	10
仮数	44
仮数の符号	45
型が違う	50
型宣言	10
かなキー	7, 165
画面ファイル	157, 158
カラーモード	27
カリキュラム例	137
関係演算	11
関数	12, 34, 150
関数定義	19
関数の自由定義	151
画面移動	27
画面プリント	49
元利合計	151
偽	11
キー	6, 164
キーボードが押されたか	36
キーボードマップ	94, 95
機械語	16, 82, 98
機械語サブルーチン	19, 41
機械語説明表	104
機械語モニター	16
基準周波数	18
基本文法	8
疑問符	10
逆アセンブラ	99
キャラクターコード表	78, 178
キャラクターモード	27
キャリッジリターン	62
キャリー	83
キャリーフラグ	83, 105
旧行番号	17
旧ファイル名	17
行数	30
強調文字	53
行の終り	87
行の抹消	14

行番号	8
行番号発生	14
空白	41
クラスター	56
グラフィックモード	27, 89, 95
クロック	41, 82
クロックパルス	82
計測器	91
桁数	30
桁数指定	42
結合	10
検索	147
検証	14
ゲタ	46
減算	10
減数カウンタ	83
コードで名前を探す	147
コール	135
コールドスタート	99
高解像度	74
交換	135
交流信号	91
コサイン	34
コマンド	8, 14, 167
コロソ	10
コントロールバス	82
コンマ	10
サービスルーチン	107
最下位8ビット	44
最小値を求める	149
最上位8ビット	44
最大の整数	34
最大のレコード番号	39
サイン	35
サインフラグ	83
索引作成	159
索引プリント	160
作表	146
サブルーチン	47, 107
算術演算	10
シークレット	26
シーケンシャルファイル	61, 62
シーケンシャルファイルの終わり	38
シーケンシャルファイルのプログラム例	64
式	9

指数	44
指数8ビット	44
指数型表示	43
システムディスク	57, 59, 73
システムプログラム	59
時刻	41
自然対数	34
実行状態の追跡	32
実行を再開	14
ジャケット	56
ジャンパー	90
ジャンパー接続	91
ジャンプ	106, 135
縮小文字	53
出力ポート	25
主レジスター	83
順編成ファイル	61
上位バイト	86
条件判定	11
乗算	10
小数点	42
小数点以下切りすて	34
剰余	10
除算	10
シリアル (直列)	91
真	11
新行番号	17
新ファイル名	17
水平タブ	54, 55
数値	9
数値関数	34
数値へ変換	37
スクリーンエリア	85
スクリーンステートメント	26
スクロール	27
スタック	85
スタックの作り方	51
スタックポインター	84
ステートメント	8
ストリング関数	36
ストリングディスクリプター	47, 48
ストリング用スタック	85
スピンドル装置孔	56
スペースキー	7
正規化	45
正規分布ヒストグラム	154
整数	10

整数変換	34, 38
正負の符号	35, 42
セクター	23, 56
接続手法	91
接続装置	91
絶対値	34, 45
セミコロ	10
セレクト	52
ゼロフラグ	83, 107
線	28
専用レジスター	83
線を描く	139
ソースフィールド	104
操作・特殊ジャンプ	136
属性	29, 58, 88
その他の関数	40, 41
ターミナルモード	90
ダイナミックRAM	84
代入	10
ダイレクトモード	8
タンジェント	35
単精度	10
単精度数値変換	34, 38
注釈	22
直流信号	91
データディスク	57
データバス	33, 82, 93, 95
データファイル	61
データを書く	25
定義	19
定数	8
ディスエーブル・インタラプト	109
ディスク	56
ディスク関数	38, 39
ディスクの構造	56
ディスクファイル	57
ディスクファイルの使用宣言	24
ディスクユニット	56
ディレクトリー	57, 58
デクリメント	105
デスティネーションフィールド	104
デバイス	92
デバイスコード	92
テレビ用アダプター	74
点が進む	142
テンキー	32, 150
転送	104

点を描く	28
等価	11
等価判定	10
特殊ステートメント	31
特殊な I/O	67
特殊文字	9
ドット	40
ドットが進む	142
ドットスペース	55
ドットを描く	28
ドライブ	23
ドライブ番号	68
トラック	23, 56
トラックの使用割り当て	57
トランスレータ	86
トレースモード	32
内蔵クロックの日付	40
内部表現	44, 45, 46
入出力	134
入出力バッファのアドレス	41
入力ポートのモニター	26
ヌルストリング	18
ノーオペレーション	108
ノーマル	26
ノンマスカブル	107
バージョン	60
ハーフキャリー	83
パイカ	53
パイカピッチ	52
倍精度	10
倍精度数値変換	34, 38
排他的論理和	11
バイナリー形式	17
配列 (添字つき変数)	19, 144
配列の大きさの計算	30
箱	27, 28
箱の中で玉が動く	147
箱を描く	140
バックアップ	59
バッファ	65, 66
パラレル (並列)	91
パリティ	83
パリティ・イーブン	106
パリティ・オッド	106
パリティ・ビット	91
番地 (アドレス)	82
ハンドアセンブル	99

汎用同期非同期送受信器	91
引き数	47
ヒストグラム	154
左回転	111
左シフト命令	111
左づめ	42
左の x 文字	36
左ローテート・アキュムレータ	111
左ローテート・サーキュラー	111
左ローテート・ディジット	112
ヒドンビット	45
ビット	152
ビデオ RAM	88
標識ラベル	56
ピリオド	10
ファイル	57
ファイル・アロケーション・テーブル (FAT)	58
ファイル削除	24
ファイルのコピー	155
ファイルの属性	39, 58
ファイルポインター	25
ファイル名	58
ファンクションキー	15
ファンクションキー定義	31
ファンクションキーの表示	27
ファンクションコード	26, 27
フィールドスイッチ	18
フィールドバッファ	38
フォーマッティング	15, 60
フォーマット	60
ブザー	31
負数	10
フック番地	101
浮動小数点	10
浮動小数点アキュムレータ	47
浮動小数点形式	44
負の符号	42
フラグ	83
ブリンク	26
プリンター	52
古いバージョン	60
ブレークポイント	101
プログラムの終わり	87
プログラムのカウンタ	84
プログラムの画面表示	15

プログラムの実行中止	23
プログラムの比較	156
プログラムを分割して実行	16
プログラムのモード	8
プログラムのロード	14
ブロック転送	107
フロッピーディスク本体	56
プロポーショナル	53
負論理	33
分類	149
米国電子工業会	91
平方根	35
べき乗	10
ヘッド・アクセス孔	56
ヘッドの位置	40
ベリファイ	16
変数	9, 144
変数のアドレス	41
変数用領域	20
変復調装置	91
ポート番号	94, 95
ポーレート	18, 90
報告書作成プログラム	162
補助レジスター	83, 84
マウントコマンド	58
マザーバス	76
マスク	26
マルチステートメント	8, 22
丸める	43
右シフト命令	112
右づめ	42
右のx文字	37
右ローテート・サーキュラー	111
右ローテート・ディジット	112
未使用クラスター	39
ミニディスクユニット	73
ミニフロッピーディスク	56
無限ループ	33
命令デコーダー	82
命令レジスター	82
メモリー	82, 84
メモリーアドレスマップ	85, 180
メモリーエリアの空き	40
メモリー増設	76
メモリー領域確保	19
モーター制御	31
文字検索	36

文字しかできない	49
文字領域	18
文字列	9
文字列サーチ	156
文字列に変換	37
文字列の長さ	36
文字をx個与える	37
モデム	90, 91
モニタープログラム	98
モニターモード	16, 49
矢印	7
ユーザー関数	47
ユーザールーチン	19
ユーザールーチンのエラー	50
余白を*でうめる	42
読み込む	16
予約語	9
予約語マップ	96, 97
ライト制御	76
ライトプロテクトノッチ	17, 56
ライトペン	74
ラインフィード	62
乱数	35, 151
ランダムに箱を描く	148
ランダムファイル	61, 65
ランダムファイルバッファ	20
ランダムファイルプログラム例	66
乱編成ファイル	61
リードアフターライト	58
リード制御	76
リアルタイム割り込み	75, 76
リスタート	135
リセットボタン	9
リターン	135
リバース	26
リバースシークレット	26
リバースブリンク	26
レコード	57
レジスター	82, 83, 84
レジスターの内容表示	101, 102
レフトマージン	55
論理演算	11
論理積	11
論理和	11
ワークエリア	85
割り込み制御	75, 76
割り込みモード	109

＜ 索 引 ＞

ABS (絶対値) -----	3 4	CPU (中央演算装置) -----	8 2
ACK (肯定応答) -----	9 2	Cレジスター -----	8 3
ADC (桁上がりこみで足す) -----	1 0 5	CPUコントロール -----	1 3 6
ADD (加算) -----	1 0 5	CPU制御 -----	8 2
ALU (演算論理機構) -----	8 2	CPU待機 -----	1 0 9
AND (論理積) -----	1 1, 3 0, 1 0 5	CPUブロック図 -----	8 2
APPEND (つけ足す) -----	2 5, 6 2	CPUレジスター -----	8 2, 8 3
AS (～として定義) -----	2 0, 2 4	CR (キャリッジリターン) -----	6 2, 8 0
ASC (文字をアスキーコード変換) -----	3 6	CSAVE (プログラム保存) -----	1 4
ATN (アークタンジェント) -----	3 4	CSNG (単精度変換) -----	3 4
ATTR\$ (属性文字を返す) -----	3 9	CSRLIN (カーソル縦の位置) -----	4 0
AUTO (行番号自動発生) -----	1 4	CTRL (コントロールキー) -----	6, 8 0
Aレジスター -----	8 3	CVD (ディスク倍精度変換) -----	3 8
BANK (バンク) -----	7 7	CVI (ディスク整数変換) -----	6 5
BASIC言語 -----	8	CVS (ディスク単精度変換) -----	6 6
BASICの内部型 -----	8 6	D (ダンプメモリー) -----	1 6
BCDコード -----	1 0 8	DAA (BCDコードに直す) -----	1 0 8
BEEP -----	3 1	DATA (データ格納場所) -----	1 8
bit -----	1 1 3	DATE\$ (日付) -----	4 0
BL (BEEP) -----	8 0	DEC (デクリメント) -----	1 0 5
Bottom -----	5 2	DEFDBL (倍精度宣言) -----	1 9
Busy (稼働中) -----	3 3, 9 2	DEFFN (ユーザー関数定義) -----	1 9
Bレジスター -----	8 3	DEFINT (整数宣言) -----	1 9
CALL (呼び出し) -----	1 0 7	DEFSNG (単精度宣言) -----	1 9
CCF (キャリーフラグ反転) -----	1 0 8	DEFSTR (文字型宣言) -----	1 9
CDBL (倍精度変換) -----	3 4	DEFUSR	
CHR\$ (アスキーコード文字変換) -----	3 6	(機械語サブルーチン) -----	1 9, 4 7
CINT (小数切り捨て整数変換) -----	3 4	DEL (デリートキー) -----	7
CL (画面クリア) -----	8 0	DELETE (行抹消) -----	1 4
CLEAR (御破算領域設定) -----	1 8, 4 7	DEレジスター -----	8 3
CLOAD (カセットテープを読む) -----	1 4	DI (割り込み禁止) -----	1 0 9
CLOAD? (セーブの検証) -----	1 4	DIM (配列宣言) -----	1 9
CLOSE (ファイル終了) -----	2 3	DISK Version -----	5 9
CLOSE# (同上) -----	6 2, 6 5	DJNZ (減算0以外ジャンプ) -----	1 0 7
CLR (画面クリアーキー) -----	7	DMA	
CN8 (ジャンパー線) -----	9 1	(ダイレクトメモリーアクセス) -----	9 2
COLOR (画面の色・機能指定) -----	2 7	DSKF (未使用クラスター) -----	3 9
Complete (終了表示) -----	5 9	DSKI\$ (セクター内容をバッファへ	
CONSOLE (画面モード設定) -----	2 7	取り出す) -----	3 8, 6 7, 6 8
CONT (継続) -----	1 4, 1 4 0	DSKO\$ (上の逆) -----	2 3, 6 7
COS (コサイン) -----	3 4	EI (割り込み可能) -----	1 0 9
CP (比較) -----	1 0 5	ELSE (IF～THEN～ELSE) -----	2 1
CPD (比較と減数) -----	1 0 8	END (実行終了) -----	2 0
CPDR (条件成立までCPD) -----	1 0 8	EOF (エンド・オブ・ファイル)	
CPI (比較) -----	1 0 8	-----	3 8, 6 2
CPIR (条件成立で実行終了) -----	1 0 8	EQV (等価) -----	1 1
CPL (Aレジを反転) -----	1 0 8	ERASE (配列抹消) -----	2 0

ERL (エラー行番号) -----	4 0	IND (入力ポートから読む) ---	1 1 0
ERR (エラー番号) -----	4 0	INDR (入力ポートから読む) ---	1 1 0
ERROR (エラーシミュレート) --	3 1	INI (入力ポートから読む) --	1 1 0
ESC (エスケープキー) -----	6	INIR (INIの反復) -----	1 1 0
EVEN (偶数パリティ) -----	1 8	INKEY\$	
EX (交換) -----	1 0 6	(キーボード監視) -----	3 6, 8 0
EXP (eのn乗) -----	3 4	INP (I/Oポートから読んだデータ	
EXX (補助レジへ交換) -----	1 0 6	の値) ---	3 3, 4 0, 1 5 0
f・1~f・10 -----	6, 7, 1 5	INPUT (入力) -----	2 4
FAC		INPUT# (入力) -----	2 4, 6 2
(浮動小数点アキュムレーター) --	4 7	INPUT\$ (入力) -----	3 6, 6 2
FAT (ファイル配置表) -----	5 7, 5 8	INS (インサート(挿入)キー) --	7
FDC I/Oポート -----	7 5	INSTR (文字検索) --	3 6, 1 4 7
FIELD (領域割当) -----	2 0, 6 5	INT (整数) -----	3 4
FILES (ファイル内容表示) -----	1 5	IPL (イニシャル・プログラム・	
FIX (小数以下切り捨て整数) ---	3 4	ロード) -----	5 7
FOR (反復条件) -----	2 0	Iレジスター -----	8 4
FORMAT (フォーマット設定) --	1 5	IX, IYレジスター -----	8 4
FPOS (ファイルセクター番号) --	3 8	JP (ジャンプ) -----	1 0 6
FRE (空きメモリー) -----	4 0, 1 3 9	JR (相対ジャンプ) -----	1 0 6
Fレジスター -----	8 3	KEY (ファンクションキー定義)	3 1
G (ジャンプ, 機械語開始) -----	1 6	KEYLIST (同上一覧表) ---	1 5
GET (ランダムファイルの読み込み)		KILL (ファイル削除) -----	2 4
-----	2 3, 6 5	Kの長さの線が進む -----	1 4 4
GET@ (図形移動) -----	2 7	L (テープから機械語を読む) ---	1 6
GOSUB		LD (ロード命令) -----	1 0 4
(サブルーチンへとぶ) -----	2 1, 2 2	LDD (ロード, 減数) -----	1 0 7
GOTO (指定行へ) -----	2 1, 2 2	LDDR (条件付きロード) ---	1 0 7
GRAPH (グラフィックキー) -----	7	LDI (ブロック転送) -----	1 0 7
HALT (CPU待機) -----	1 0 9	LDIR (条件付ブロック転送)	1 0 7
HEX\$ (16進変換) -----	3 6	LEFT\$ (左から文字検索) ---	3 6
HM (ホームポジション) -----	8 0	LEN (文字の長さ) -----	3 6
HOME (ホームポジションキー) --	7	LET (変数に値代入) -----	2 1
HT (水平タブ) -----	8 0	LF (ライン・フィード) --	6 2, 8 0
HLレジスター -----	8 3	LFILES	
I/Oアドレス -----	4 0	(ディスクファイル表示)	1 5, 5 2
I/Oポート -----	8 2	LINE (線, 箱を描く) 2 7, 1 4 0	
I/Oマップ -----	9 2, 9 3	LINEINPUT (一括入力) --	2 4
I/Oユニット -----	7 3, 7 7	LINEINPUT#	
ID (識別子) -----	5 7	(ディスク用一括入力) --	2 4, 6 2
IDの初期化 -----	1 5	LIST (プログラム画面表示) --	1 5
IEEE-488 -----	7 5, 9 1	LLIST	
IF (条件分岐) -----	2 1	(同上プリンター出力) --	1 5, 5 2
IM0~2 (割り込みモード) -----	1 0 9	LOAD (ディスクからロード) --	1 6
IMP (インプリケーション) -----	1 1	LOC (レコード番号を) ---	3 8, 6 5
IN (入力命令) -----	1 0 9, 1 1 0	LOCATE (カーソルの位置) --	2 8
INC (インクリメント) -----	1 0 5	LOF (最大レコード番号)	3 9, 6 5

LOG (自然対数) ----- 3 4
 LPOS
 (プリンターヘッドの位置) 4 0, 5 2
 LPRINT (帳表印刷) ----- 2 4, 5 2
 LPRINTCHR\$ (1 2)
 (プリンターの改ページ) ----- 5 2
 LPRINT USING
 (出力帳票編集) ----- 5 2
 LSET (左づめ) ----- 2 1, 6 5, 6 6
 LV (セーブした機械語ベリファイ) 1 6
 MAKINT (整数ルーチン) ----- 5 0
 MERGE
 (プログラムの混ぜ合わせ) ----- 1 6
 MID\$ (文字検索) ----- 3 6
 MKD\$ (文字列変換) 3 9, 6 5, 6 6
 MKI\$ (文字列変換) 3 9, 6 5, 6 6
 MKS\$ (文字列変換) 3 9, 6 5, 6 6
 MOD (剰余) ----- 1 0
 MON (モニターモードにする) ----- 1 6
 MOTOR (カセットモーター制御) 3 1
 MOUNT (ディスクを設定する)
 1 7, 5 9, 6 2, 6 5
 NAME (ファイル名変更) ----- 1 7
 NEG (2の補数) ----- 1 0 8
 NEW (プログラムと変数クリア) 1 7
 NEXT (ループの範囲設定) ----- 2 1
 NON (Nonパリティ) ----- 1 8
 NOP (ノーオペレーション) ----- 1 0 8
 NOT (否定) ----- 1 1, 3 0
 n行改行 ----- 5 5
 OCT\$ (8進変換) ----- 3 7
 ODD (奇数パリティ) ----- 1 8
 ON (条件分岐) ----- 2 2
 ON ERROR GOTO
 (エラー発生時の分岐) ----- 3 1
 OPEN (ファイル使用宣言) ----- 2 4
 OPEN# (ファイルを開く) 6 2, 6 5
 OR (論理和) ----- 1 1, 3 0
 OTDR (出力命令) ----- 1 1 0
 OTIR (出力命令) ----- 1 1 0
 OUT (出力命令) ----- 2 5, 1 1 0
 OUTD (出力命令) ----- 1 1 0
 OUTI (出力命令) ----- 1 1 0
 OUTPUT (出力ほか) ----- 2 5, 6 2
 PC-6000シリーズ ----- 1 6 3
 PC-8000シリーズラインアップ 7 3
 PEEK (番地をのぞく) ----- 4 0, 4 6

PLAY (音出し) ----- 1 6 8
 POINT (ドットの有無) ----- 4 0
 POKE (データ書き込み) 2 5, 4 8
 POP (スタックから取り出す) 1 0 5
 POS (カーソルの横の位置) ----- 4 1
 PRESET (ドット消去) 2 8, 3 0
 PRINT (画面表示) ----- 2 5
 PRINT#
 (ファイルにデータ記入) ----- 2 5
 PRINT# USING
 (指定書式でレコード書き出し) 6 2
 PRINT CHR\$
 (文字や図形を表示する) ----- 8 0
 PRINT USING
 (画面出力編集) ----- 2 5, 4 2, 4 3
 PRINT# (レコード書き出し) 6 2
 PROM (プログラマブルROM) 8 4
 PSET (ドット表示) ----- 2 8, 3 0
 PUSH (スタックへ入れる) ----- 1 0 4
 PUT (レコードを書く) ----- 2 6, 6 5
 PUT@ (図形移動) ----- 2 9
 PUT@する画面の状態 ----- 3 0
 PCレジスター ----- 8 4
 RAM
 (ランダムアクセスメモリー) ----- 8 4
 READ (データを読む) ----- 2 2
 Ready (待機中) ----- 3 3
 READとDATA ----- 1 4 2
 REM (注釈行) ----- 2 2
 REMOVE
 (終了作業) ----- 1 7, 6 2, 6 5
 RENUM (行番号つけかえ) ----- 1 7
 RES (リセットビット) ----- 1 1 4
 RESTORE (回復) ----- 2 2, 1 4 3
 RESUME (再開) ----- 3 2
 RET (リターン命令) ----- 1 0 7
 RET (RETURNキー) ----- 7
 RETURN (サブルーチン終了) 2 2
 RETURNキー ----- 7
 RIGHT\$ (右から文字検索) ----- 3 7
 RLA (左ローテート・アキュムレータ
 ー) ----- 1 1 1
 RLCA (左ローテート・サーキュラー
 ・アキュムレーター) ----- 1 1 1
 RLD (左ローテート・ディジット・ア
 キュムレーター) ----- 1 1 2
 RND (乱数) ----- 3 5, 1 5 1

ROM		STOPキー	7
(リードオンリーメモリー)	8 4	STR\$ (数値を文字列変換)	3 7
RRA (右ローテート・アキュムレータ		STRING\$ (文字列をあたえる)	3 7
ー)	1 1 1		
RRCA (右ローテート・サーキュラー		SUB (減算)	1 0 5
・アキュムレーター)	1 1 1	SWAP (内容交換)	2 3, 1 4 9
RRD (右ロヒテート・ディジット・		TAB (タブ)	4 1
アキュムレーター)	1 1 2	TAN (タンジェント)	3 5
RS-232C	7 5, 9 1	TERM (ターミナルモード)	1 8, 9 0
RSET (右づめで)	2 3, 6 5, 6 6	THEN (条件分岐)	2 1
RST (リスタート)	1 0 7	TIME\$ (時刻)	4 1, 1 3 9
RUN (実行)	1 7	TM (テストメモリー)	1 6
Rレジスター	8 4	TO (FOR~TO)	2 0
S (セットメモリー)	1 6	Top of form	5 2
SAVE (ディスクへ保管する)	1 7	TROFF (トレースモード解除)	3 2
SBC (桁下げこみ減算)	1 0 5	TRON (トレースモード)	3 2
SCF (セットキャリーフラグ)	1 0 8	USART	
SCREEN (画面)	1 6 9	(汎用同期/非同期送受信器)	9 1
SEL (セレクトスイッチ)	5 2	USR (ユーザー関数)	4 1
SET (ファイルの属性)	1 8	USR関数	4 7
SET (セットビット)	1 1 4	VAL (数値変換)	3 7
SGN (正負符号を与える)	3 5	VARPTR (バリエブルポインター)	4 1, 4 6
SIN (サイン)	3 5	W (テープへセーブする)	1 6
SLA (左シフト命令)	1 1 1	WAIT	
SOUND (音を作る)	1 6 7	(入力ポートモニター)	2 6, 3 2
Sレジスター	8 4	WIDTH (画面設定)	3 0
SPACE\$ (空白を与える)	3 7	XOR (排他的論理和)	1 1, 3 0
SPC (プリント文中空白を与える)	4 1	x個の空白	3 7, 4 1
SQR (平方根)	3 5	x軸, y軸	1 4 1
SRA (右シフト命令)	1 1 2	x番地の内容	4 0
SRL (右シフト命令)	1 1 2		
STEP (増分)	2 0		
STOP (実行中止)	2 3, 1 4 0		

パソコンPCシリーズ ハンドブック

昭和57年3月 1日 第1刷
 昭和57年6月25日 第7刷
 編者 朝日新聞社電子計算室
 発行者 波多野公介
 印刷製本 凸版印刷株式会社
 発行所 朝日新聞社
 ①104 東京都中央区築地5-3-2
 電話 03 (545) 0131 (代表)
 編集=出版局プロジェクト室 販売=出版販売部
 振替 東京0-1730
 ©朝日新聞社1982 2055-258141-0042

定価800円

パソコンPCシリーズ 8001 6001 ハンドブック

●本書の主な内容

1. PC8001 キー配置図
2. BASICの基本文法
3. 操作上の注意点
4. 文字・演算子一覧表
5. コマンドとステートメント一覧
6. 数値関数
7. スtring関数
8. ディスク関数
9. その他の関数
10. PRINT USING 一覧表
11. 浮動小数点形式
12. USR関数
13. プリンターの使い方
14. ミニフロッピーディスク
15. エラーメッセージ
16. PC8000 ラインアップ
17. キャラクターコード表
18. PRINT CHR\$ INKEY\$
19. CPUブロック図
20. CPUレジスター
21. メモリーアドレスマップ
22. BASICの内部型
23. ビデオRAM
24. ターミナルモード
25. I/Oマップ
26. キーボードマップ
27. 予約語マップ
28. 機械語プログラムの作り方
29. Z80相当機械語説明表
30. 機械語ニーモニック変換表
31. ニーモニック機械語変換表
32. カリキュラム例
33. 講習テキスト15回分
34. ディスクを使ったプログラムの例
35. PC6001 機能解説
キー配置図 文字・演算子
コマンド・ステートメント一覧表
サウンド・プレー・カラー
キャラクターコード表
メモリーアドレスマップ
エラーメッセージ
36. 索引